UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Manipulación digital de prototipos en papel

AUTORES: Iriarte Bosnic, Marcos Matias – Taus, Johana Giselle

DIRECTOR: Julián Grigera

CODIRECTOR: José Matias Rivero

ASESOR PROFESIONAL:

CARRERA: Licenciatura en Sistemas

Resumen

Esta tesina busca revalorizar la utilización de prototipos en papel como una herramienta que puede ser de gran utilidad en las primeras etapas del diseño y desarrollo de un producto de software. Mediante los prototipos en papel podemos obtener un primer acercamiento al producto que necesitamos construir y así obtener de manera rápida una primera aproximación a los requerimientos finales del desarrollo. Por este motivo, mediante la manipulación digital de los prototipos en papel buscamos poder ofrecer nuevas funcionalidades a una herramienta tan sencilla, pero a la vez útil como son los prototipos en papel.

Palabras Clave

Wireframes, Prototipos en papel, prototipos digitales, usabilidad, evaluación de usabilidad, elicitación de requerimientos, mockups, métricas de usabilidad, evaluación de productos de software, reducción de errores en etapas tempranas del desarrollo.

Conclusiones

Se consiguió desarrollar una aplicación que permite la digitalización de prototipos en papel. Ésta hace posible añadir nuevas funcionalidades como la visualización e interacción con los prototipos digitalizados, y hace posible realizar pruebas de usabilidad para obtener métricas. Con estas métricas es posible evaluar usabilidad directamente sobre los prototipos digitalizados.

Trabajos Realizados

Desarrollo de una aplicación móvil que contribuya con la elicitación de requerimientos y permita digitalizar prototipos de productos de software desarrollados en papel. Creación de un mecanismo para poder realizar pruebas de usabilidad sobre los prototipos digitalizados y así, de esta manera, poder obtener diferentes métricas de usabilidad que nos permitan evaluar distintos aspectos de usabilidad de manera temprana en las primeras etapas de diseño de un producto de software.

Trabajos Futuros

Mejorar la digitalización de los prototipos mediante técnicas de detección automática de los diferentes componentes de los prototipos. Añadir más widgets a los ya creados en la aplicación. Permitir la edición colaborativa de los prototipos digitalizados. Crear mecanismos independientes para poder ejecutar las pruebas de usabilidad en diferentes dispositivos. Agregar nuevas métricas y parámetros que nos permitan obtener mayor información sobre potenciales problemas de usabilidad.

Fecha de la presentación: Septiembre 2021

UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Manipulación digital de prototipos en papel

AUTORES: Iriarte Bosnic, Marcos Matias – Taus, Johana Giselle

DIRECTOR: Julián Grigera

CODIRECTOR: José Matias Rivero

ASESOR PROFESIONAL:

CARRERA: Licenciatura en Sistemas

Resumen

Esta tesina busca revalorizar la utilización de prototipos en papel como una herramienta que puede ser de gran utilidad en las primeras etapas del diseño y desarrollo de un producto de software. Mediante los prototipos en papel podemos obtener un primer acercamiento al producto que necesitamos construir y así obtener de manera rápida una primera aproximación a los requerimientos finales del desarrollo. Por este motivo, mediante la manipulación digital de los prototipos en papel buscamos poder ofrecer nuevas funcionalidades a una herramienta tan sencilla, pero a la vez útil como son los prototipos en papel.

Palabras Clave

Wireframes, Prototipos en papel, prototipos digitales, usabilidad, evaluación de usabilidad, elicitación de requerimientos, mockups, métricas de usabilidad, evaluación de productos de software, reducción de errores en etapas tempranas del desarrollo.

Conclusiones

Se consiguió desarrollar una aplicación que permite la digitalización de prototipos en papel. Ésta hace posible añadir nuevas funcionalidades como la visualización e interacción con los prototipos digitalizados, y hace posible realizar pruebas de usabilidad para obtener métricas. Con estas métricas es posible evaluar usabilidad directamente sobre los prototipos digitalizados.

Trabajos Realizados

Desarrollo de una aplicación móvil que contribuya con la elicitación de requerimientos y permita digitalizar prototipos de productos de software desarrollados en papel. Creación de un mecanismo para poder realizar pruebas de usabilidad sobre los prototipos digitalizados y así, de esta manera, poder obtener diferentes métricas de usabilidad que nos permitan evaluar distintos aspectos de usabilidad de manera temprana en las primeras etapas de diseño de un producto de software.

Trabajos Futuros

Mejorar la digitalización de los prototipos mediante técnicas de detección automática de los diferentes componentes de los prototipos. Añadir más widgets a los ya creados en la aplicación. Permitir la edición colaborativa de los prototipos digitalizados. Crear mecanismos independientes para poder ejecutar las pruebas de usabilidad en diferentes dispositivos. Agregar nuevas métricas y parámetros que nos permitan obtener mayor información sobre potenciales problemas de usabilidad.

Fecha de la presentación: Septiembre 2021

UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE GRADO

Manipulación Digital de Prototipos en Papel

Autores:

Iriarte Bosnic, Marcos Matias

Taus, Johana Giselle

Director:

Julián Grigera

Co-Director:

José Matias Rivero

Septiembre 2021

Agradecimientos

A nuestros directores, Julián y Matías, quienes nos han enseñado, acompañado y ayudado a construir las alas que hoy en día nos dejan ser y hacer lo que más nos gusta. Sin su compañía durante el desarrollo de esta tesina no hubiera sido lo mismo.

Agradecemos a nuestras familias: por incentivar desde pequeños nuestros gustos por la informática, por brindarnos las herramientas necesarias para crecer personal, académica y profesionalmente. Por el esfuerzo incondicional para cuidar y priorizar siempre nuestro crecimiento personal, por sus esfuerzos para permitirnos completar nuestros estudios.
"Habremos de ser lo que hagamos, con aquello que hicieron de nosotros" (Sartre).

Por último, agradecemos a la Facultad de Informática y a la Universidad Nacional de La Plata por la formación dada, a cada uno de nosotros. A todos aquellos que eligen y dan sustento a un modelo de educación pública, gratuita y de calidad. Es difícil imaginarnos en esta misma posición si nuestra sociedad no hubiese garantizado este derecho, incluso desde la más temprana de las etapas.

Índice General

Índice de figuras

Capítulo 1: Introducción

Este proyecto busca revalorizar los wireframes en papel para así facilitar el trabajo de los diseñadores y desarrolladores. Un **wireframe** es una representación visual de una interfaz de usuario baja fidelidad. Realizar un wireframe en papel nos permite expresar las ideas de forma concreta, y gracias a ello se reducen errores y se aclaran los requerimientos. Éstos nos ayudan a pensar estratégicamente el recorrido del usuario y el vínculo que se pretende obtener con una interfaz de usuario, así como potenciar la resolución de las necesidades que este tenga.

Sumergiremos a los lectores en los diferentes problemas que podrían ocurrir al tener fallas de usabilidad en nuestros productos de software, como así también pondremos énfasis en la importancia de realizar una buena elicitación de requerimientos al momento de comenzar un desarrollo de un programa informático.

Por otro lado, brindaremos los conceptos básicos necesarios para poder comprender cada uno de los capítulos que se presentan en esta tesis, así como introducir al lector en la problemática que intentamos resolver.

En este capítulo se plantean los objetivos de la tesina, la motivación que nos incentivó a realizar la misma buscando hacer aportes en el marco de la usabilidad y técnicas de elicitación de requerimientos.

En la sección **1.5 Conceptos introductorios** se definirán términos generales e indispensables de conocer antes de proceder con la lectura, como por ejemplo wireframes, mockups, prototipos o paper prototyping, entre otros. Si el lector ya conoce estos conceptos podrá continuar con la lectura sin necesidad de leer esta sección, la cual sirve de referencia en caso de necesitar reforzar algún tema en particular.

1.1 Motivación

El avance tecnológico de los dispositivos móviles ha ido creciendo rápidamente en los últimos años. Cada año nace una nueva generación de *smartphones* y tabletas incorporando nuevas tecnologías (GPS, lector de huellas, notificaciones, acelerómetro, cámara, etcétera). Éstas permiten obtener mayor información para adicionar nuevas funcionalidades, mejorar la experiencia de usuario y potenciar la comunicación. De este modo, las aplicaciones móviles (apps) permiten establecer un buen canal de transmisión entre una entidad y el usuario o entre los distintos usuarios (que comparten un interés en común).

Desde el punto de vista de la ingeniería de software, el proceso de desarrollo de estas aplicaciones móviles requiere agilidad y flexibilidad. Por lo general, la obtención de requisitos es un proceso que demanda tiempo y tiene un cierto grado de dificultad, durante esta etapa de ingeniería de requisitos es importante que se puedan obtener y recolectar la mayor cantidad de requerimientos posible. Para ello, se puede utilizar un método de prototipo de papel desechable y accesible, así como el contenido dinámico que suelen consumir diariamente los usuarios.

La técnica de paper prototyping (prototipado en papel) no es una novedad y se utiliza hace tiempo para el análisis y primera etapa de desarrollo (Holzmann, 2012). Esta técnica permite diseñar, probar, refinar la interfaz de usuario y erradicar problemas de usabilidad. Además, cabe destacar que es una técnica ampliamente utilizada durante las primeras etapas del diseño de la interfaz de usuario (Li, 2010). Permite que un diseñador reciba comentarios tempranos del usuario sobre un diseño. Un diseñador puede crear una maqueta de interfaz utilizando dibujos a mano y probar una idea inicial con un usuario en un forma visual y tangible.

Siguiendo con las técnicas en etapas tempranas del diseño, se utilizó el término de "diseño de interacción" para describir el papel que faltaba en el desarrollo de software, y además se menciona "Casi todo diseño de interacción se refiere a la selección de comportamiento, función e información y su presentación a los usuarios" (A. Cooper, 2004).

Según Snyder (Carolyn Snyder, 2003), los beneficios más importantes de la creación de prototipos en papel consisten en que permiten recopilar comentarios de

los usuarios antes de que comience la implementación real, facilitando el desarrollo iterativo rápido, y no requieren de ninguna habilidad técnica.

La creación de prototipos en papel se puede integrar fácilmente en el proceso de diseño, ya que la mayoría de los diseñadores prefieren dibujar en las primeras etapas. Las principales razones de la creación generalizada de bocetos es que son rápidos de producir, el uso del papel y lápiz es natural, y los bocetos evitan centrarse en detalles sin importancia. Esta clase de prototipos suelen denominarse prototipos de "baja fidelidad", en contraste con los de "alta fidelidad" que suele estar más cerca de los diseños finales. Sin embargo, esta no es la única categorización posible. Kolko, en su libro *Thoughts on interaction design* comenta que "La creación de prototipos comúnmente se define como de baja o alta fidelidad, pero este conjunto de descriptores es muy superficial en el rango de posibilidades" (Kolko, J., 2010). En *"Breaking the Fidelity Barrier"* (McCurdy, M. et al. 2006) se describen cinco dimensiones a lo largo de las cuales la fidelidad puede variar, de menor a mayor:

- Fidelidad visual
- Interactividad
- Profundidad
- Amplitud
- Fidelidad de datos

Para este trabajo, nos focalizamos en las dos primeras. Según los autores, estas dimensiones capturan la importancia del nivel de fidelidad visual de los prototipos. Para ello podemos partir de modelos poco fieles, como bocetos dibujados a mano, hasta maquetas con alto nivel de detalle. Según McCurdy, un prototipo visualmente detallado podría generar evaluaciones a nivel estético en lugar de evaluaciones de funcionalidad y usabilidad. Es por ello que hemos decidido concentrarnos en la técnica de paper prototyping, ya que al contar con bajo nivel de detalles estéticos nos simplifica el análisis de funcionalidad y usabilidad sobre los prototipos.

Realizar un wireframe en papel nos permite expresar las ideas de forma concreta para que otras personas logren interpretarnos. Gracias a ello se reducen los errores, se aclaran los requisitos y se permite pensar estratégicamente el recorrido del usuario y el vínculo que se pretende obtener con el mismo, así como potenciar la resolución de las necesidades que el usuario tenga.

Se eligió la técnica del paper prototyping ya que consideramos el trabajo manual como una producción que muchos profesionales del campo de la informática y el

diseño utilizan en una primera instancia. A la hora de poner en común los requerimientos y funcionalidades que pretende el cliente, se suelen realizar prototipos de forma manual.

De esta manera, se reconoció la necesidad de la digitalización de estos prototipos o wireframes ya que, por su simplicidad, los prototipos de papel no sirven para realizar evaluaciones detalladas del diseño. No pueden simular la respuesta del sistema y, además, en el momento de evaluarlo es fácil que se den por supuestas cosas que realmente no están en el prototipo, entre otras razones. Realizar un prototipo de papel mientras se definen ideas y luego digitalizarlas con una simple foto que se toma desde un dispositivo móvil, nos brindaría la libertad que no tenemos mientras hacemos los mockups en la computadora.

En esta línea, nuestra tesis consiste en un esquema híbrido en el cual se combina la construcción manual de mockups con procesos automáticos de digitalización y asistencia para la detección de problemas tempranos de usabilidad.

1.2 Objetivos generales

Este proyecto busca fomentar el uso y diseño de wireframes en papel, ofreciendo su digitalización para aprovechar las ventajas del prototipado digital. Consideramos a los prototipos digitales una herramienta de gran importancia para las primeras etapas de diseño, por sus características de detección de errores o posibilidades de optimización de los prototipos.

El objetivo principal de este trabajo es el desarrollo de una metodología para evaluación temprana de usabilidad junto con una aplicación móvil, llamada MWP, que facilite la digitalización de wireframes realizados con la técnica de paper prototyping.

1.3 Objetivos específicos

Tareas a realizar para desarrollar este trabajo:

- Digitalizar wireframes realizados con la técnica paper prototyping, identificando los diferentes componentes que estén presentes en los wireframes provistos.

- Permitir a los diseñadores generar pruebas de usabilidad fácilmente sobre los wireframes digitalizados.

- Desarrollar un sistema de métricas que nos permita hacer evaluaciones de usabilidad en los prototipos digitalizados.

Entre las principales ventajas se encuentran: rapidez de carga, imágenes de alta calidad, tecnologías y optimización de contenido.

1.4 Aportes de la tesina

A continuación, se presenta una lista donde se describen, en líneas generales, los aportes realizados por la presente tesina:

- Desarrollo de una metodología para evaluación temprana de usabilidad en prototipos de productos de software.

- Asistencia en el proceso de evaluación de usabilidad a través de la utilización de una herramienta que permite generar pruebas de usabilidad sobre prototipos en papel con posibles usuarios finales.

- Digitalización de los wireframes realizados con la técnica de paper prototyping. Luego de la captura y procesamiento de imágenes, se podrían generar los prototipos correspondientes a cada uno de los wireframes que fueron capturados con la aplicación.

- Contribuir a mejorar la evaluación de interfaces de usuario mediante la creación de diferentes versiones de interfaz para una misma aplicación.

- Permitir realizar pruebas de usabilidad con usuario sobre diferentes versiones de prototipos de interfaz. De esta forma, se podrían capturar potenciales errores durante la etapa de elicitación de requerimientos, donde es más rápido corregirlos, reduciendo potenciales costos adicionales en etapas posteriores del desarrollo.

- Fomentar la utilización de sketches, wireframes y mockups como primera etapa del ciclo de vida de un producto de software. Lo ideal es que estas etapas preliminares contribuyan a encontrar cualquier problema de requerimientos o usabilidad mientras está todavía en fase inicial y así ajustar los diseños, estructura, funcionalidad o requerimientos para así poder crear productos de acuerdo a las necesidades para las cuales fueron pensados.

1.5 Conceptos introductorios

Para poder avanzar y comprender correctamente las siguientes partes de la tesina, es necesario introducir algunos conceptos básicos, los cuales mencionan y usan a lo largo de todo el desarrollo de la misma. Por ese motivo, brindaremos las definiciones de aquellos términos necesarios para poder realizar la lectura sin problemas.

1.5.1 Wireframe

Como bien expone el paper *"Automatically Generating Web Page From A Mockup"* (Huang, Long y Chen, 2016) un **wireframe** es una representación visual de una interfaz de usuario de baja fidelidad, en escala de grises, la cual se usa en las primeras etapas del proceso de desarrollo con el fin de establecer la estructura y funciones básicas antes de agregar el contenido y el diseño visual.

Murillo et. al., en su paper *"Use of wireframes and mockup on the redesign of a university website using the methodology User Centered Design"* define a los wireframe como diagramas esquemáticos y "sketches" que definen una Página Web o contenido de la pantalla y el flujo de navegación.

Los wireframes pueden ser creados en papel, directamente en HTML/CSS o con aplicaciones de software. Al ser representaciones tan simples, permiten crear y explorar múltiples versiones de un mismo proyecto de una forma rápida considerando alternativas de la interfaz de usuario (UI) y la experiencia de usuario (UX), así como aplicar cambios o nuevas ideas que aparezcan a posteriori en wireframes a papel. Pueden proporcionar información importante sobre la experiencia del usuario, detectar errores y ayudar a optimizar el diseño de la experiencia de usuario en una fase temprana de la fase de diseño.

1.5.2 Mockup

Un mockup o maqueta es un diseño digital de una web y/o aplicación. Las maquetas se utilizan para visualizar ideas y conceptos en el contexto del diseño web e incluyen la estructura de navegación, el sitio y los elementos de diseño en detalle. Los mockups se caracterizan por tener una media-alta fidelidad y por ser representaciones completamente estáticas del diseño visual. Es esencialmente eso, el diseño más visual.

Pueden ser plantillas producidas con programas de edición de imágenes sin funcionalidad o diseños que se crean con herramientas especiales de maquetas y donde los elementos de control ya están vinculados con funciones simples. Por lo tanto, su objetivo es demostrar cómo se van a representar visualmente los elementos definidos, por ejemplo, en el wireframe.

1.5.3 Prototipo

Los prototipos son representaciones de media-alta fidelidad que incluyen o simulan la interacción con la interfaz. En esta representación los usuarios ya sí podrán experimentar en alguna medida la experiencia de uso del producto. Si el wireframe define la estructura y el mockup cómo es visualmente, el prototipo define sobre todo cómo se comporta el producto. Por ello, aquí la interacción debe estar ya muy definida.

Como mencionamos con anterioridad sobre el paper del autor Li (2010) los prototipos pueden ser realizados con herramientas digitales, como Balsamiq[1]. A menudo requieren que los diseñadores renuncien a la práctica familiar de la creación de prototipos en papel o achicar manualmente la brecha entre el papel y la creación de prototipos electrónicos.

A su vez tenemos la posibilidad de agregarle interacción a los prototipos con aplicaciones como InVision[2] o Zeplin[3] que, siendo plataformas de automatización, nos permiten compartir y comunicar ideas.

---

[1] https://balsamiq.com/
[2] https://www.invisionapp.com/
[3] https://zeplin.io/

1.5.4 Paper prototyping

El paper prototyping es una técnica de prototipado rápido de baja fidelidad, como se muestra en la Figura 1, la cual se centra en encontrar el mejor diseño para los usuarios a los cuales va dirigido el sistema (*target users*). Esta técnica permite que un diseñador reciba comentarios tempranos del usuario sobre un diseño, y se utiliza hace mucho tiempo para el análisis y primera etapa de desarrollo (Holzmann, 2012).

Un diseñador puede crear una maqueta de interfaz utilizando dibujos a mano y probar una idea inicial con un usuario en un forma visual y tangible.

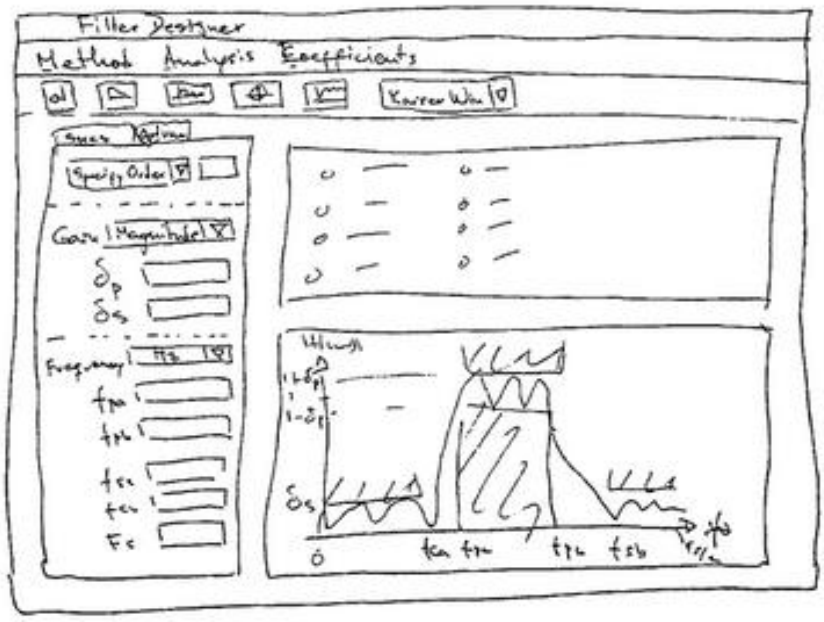

Figura 1 Prototipo en papel, hecho a mano de una vista de una aplicación de filtros para reportes científicos.

1.5.5 Storyboards

El concepto de storyboard es una serie de imágenes que representa el comportamiento de la interfaz con respecto a la resolución de una tarea en particular.

Como se explica en *Paper prototyping: The fast and easy way to design and refine user interfaces* (Snyder C. 2003) algunos storyboards como los que están en la siguiente Figura 2 incluyen las representaciones de la interfaz de usuario, pero otros storyboards son más conceptuales y de un nivel más alto.

En general son utilizados para entender el flujo de trabajo de los usuarios y conocer cómo la interfaz va a actuar frente a cada uno de los diferentes pasos. Son solamente utilizados en las etapas tempranas de desarrollo ya que los destinatarios finales de estos storyboards no pueden interactuar con estos, dado que se limitan a ser solamente representaciones gráficas de las posibles interacciones con un producto de software.

No se los clasifica como prototipos en papel, aunque eventualmente se puede seleccionar una vista y separarla del modelo para agregarle la información necesaria para darle soporte a una funcionalidad.

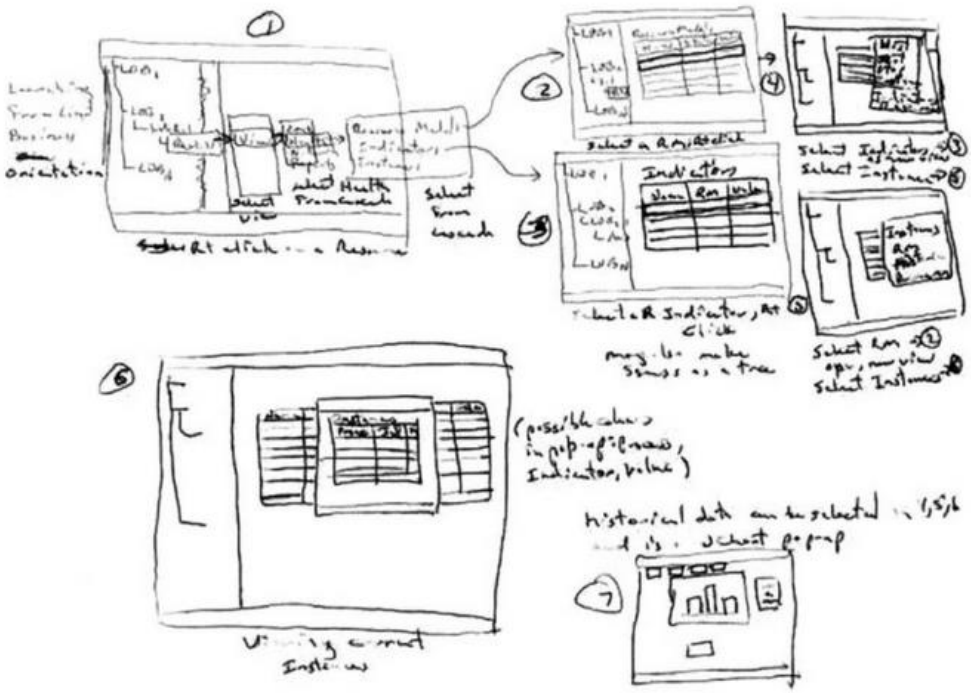

Figura 2 Storyboard a mano alzada que utilizan los desarrolladores para elaborar una secuencia de pantallas. Las anotaciones explican el flujo del sistema.

1.5.6 Usabilidad

La usabilidad de una aplicación de software se refiere a la facilidad con que los usuarios pueden utilizar la misma para alcanzar un objetivo concreto. Este nivel de usabilidad no puede medirse o ser evaluado directamente, debido a que depende de diferentes factores.

Formalmente, la definición más utilizada o reconocida de usabilidad es la que se expone en la norma **ISO 9241-11**[4], en la cual se define como el grado con el que un producto puede ser usado por usuarios específicos para alcanzar objetivos específicos con efectividad, eficiencia y satisfacción, en un contexto de uso específico.

1.5.7 Métrica

Una métrica (medida) es un valor numérico o nominal asignado a características o atributos de un objeto. Son computados a partir de un conjunto de datos observables y consistentes con la intuición, como afirma *DeMarco* en el paper *Controlling software projects: Management, measurement, and estimates* (DeMarco, T., 1986).

La validez de las métricas es corroborada a través del uso de los métodos empíricos. A continuación, se describen las características principales del método empírico utilizado en la presente tesina:

Métodos Empíricos: Este tipo de pruebas se desarrollan para casos en los cuales los desarrolladores intentan simular las condiciones reales bajo las cuales se usa la aplicación. Se le solicita al usuario que realice tareas previamente definidas, relacionadas con la funcionalidad de la aplicación. La interacción del usuario con la aplicación es evaluada para detectar los problemas que experimentó el usuario con el uso de la misma.

Para realizar la evaluación se necesita analizar la información de usabilidad que se desprende de la interacción. Para capturar esa información, por un lado, están las métricas que se van obteniendo en tiempo real al interactuar el usuario con la aplicación. Por otro lado, se puede registrar la forma de realizar las tareas por parte del usuario mediante una grabación de video y audio; también se pueden efectuar entrevistas o cuestionarios.

La ventaja de este tipo de pruebas es que directamente se monitorea el uso de la aplicación por parte de usuarios reales. La desventaja es que se deben realizar sobre aplicaciones implementadas casi en su totalidad, lo que implica que cualquier

---

[4] ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) – part 11: Guidance on usability.

modificación que se tenga que realizar demandará mayor tiempo, asegura el autor del paper *A usability-evaluation metric based on a soft-computing approach*. (Chang, E., & Dillon, T. S., 2006). Si la aplicación es un prototipo, los resultados de las pruebas son utilizados para mejorar la versión final de la misma.

1.5.8 Diseño Mobile First

El diseño *Mobile First* es un paradigma que defiende que la manera en que deberíamos pensar la interfaz de nuestro producto, servicio o empresa es al revés a la manera en que estamos acostumbrados (desde la pantalla de una computadora, versión web), es decir, comenzando por las pantallas pequeñas de los teléfonos celulares.

Primero, pensar en aquellos contenidos y funcionalidades que realmente son vitales para el usuario y eliminar todo lo demás. Así se puede realizar una buena interfaz móvil. Una vez hecha esta interfaz, se puede pensar en la adaptación a pantallas más grandes incluyendo contenido no esencial.

Los defensores de esta metodología dicen que pensar en la pantalla del celular obliga a focalizarse en lo que es realmente relevante y lo que no. Y ese ejercicio hará que el resultado final dé una mucho mejor experiencia de usuario.

Esta adaptación a teléfonos celulares no solamente significa que no haya que hacer zoom para leer los textos de la web, sino que todos los elementos deben adaptarse correctamente: botones, links, imágenes, videos y, por supuesto, los textos. Por otra parte, hay que tener en cuenta que hoy día existe una amplia cantidad de tamaños de pantallas en los dispositivos y con posibilidad de visualizar el contenido en forma vertical y horizontal. El diseño mobile first debería asegurar que se adapte el contenido a todas estas posibilidades.

Capítulo 2: Trabajos Relacionados

En este capítulo examinaremos brevemente las principales investigaciones sobre aplicaciones para el prototipado digital, como así también aplicaciones que generan el código de una aplicación web/móvil a partir de un prototipo en papel e identificaremos los problemas más comunes que surgen en dichas plataformas.

Estudiaremos las dos áreas principales, convertir dibujos de baja fidelidad (digitales o analógicos) en código y convertir wireframes de alta fidelidad en código. Ambas áreas comparten un objetivo en común - convertir un diseño en una aplicación - utilizando diversas técnicas y/o herramientas que nos permitirán realizar diseños creativos y similares a cómo quedará la aplicación una vez culminada la etapa de desarrollo. La principal diferencia es la fidelidad de los diseños que se traducen, lo que conlleva diferentes desafíos.

Hoy en día disponemos de una amplia variedad de aplicaciones colaborativas y de prototipado digital para utilizar, en el apartado de 2.1 Aplicaciones relacionadas mencionaremos las herramientas más utilizadas, por ejemplo: Balsamiq, InVision, etc., por los diseñadores al momento de crear wireframes o prototipos, junto con los beneficios que nos proporciona cada una de estas y las diferentes características que las hacen únicas en su función.

También nos enfocaremos en describir y analizar las dos áreas previamente mencionadas, sin entrar en detalle sobre campos como wireframe augmentation mencionados en el paper "*SketchWizard: Wizard of Oz prototyping of pen-based user interfaces*" (Davis, R. C., Saponas, T. S., Shilman, M., & Landay, J. A., 2007).

2.1 Aplicaciones relacionadas

Cuando buscamos aplicaciones de prototipado digital encontramos muchas opciones muy diferentes en el mercado. Desde aplicaciones bastante simples que nos permiten realizar prototipos de baja fidelidad hasta programas muy complejos, generalmente comerciales, con una infinidad de funciones y opciones para poder representar los prototipos con un alto nivel de fidelidad.

Algunas de estas aplicaciones nos permiten realizar prototipos interactivos y otras no. Existen programas que son capaces de convertir wireframes en papel en wireframes y prototipos digitales. Incluso existen algunas aplicaciones que mediante algoritmos de deep learning e inteligencia artificial nos permiten obtener el código de nuestros prototipos en papel.

En función de las diferentes características y opciones con las que cuenta cada una de estas aplicaciones, elegimos clasificarlas en cuatro grandes grupos según las características de los resultados obtenidos:

1. **Prototipos de baja fidelidad.**
2. **Prototipos de alta fidelidad sin interacción.**
3. **Prototipos de alta fidelidad con interacción.**
4. **Herramientas que convierten prototipos que generan código.**

2.1.1 Prototipos de baja fidelidad

En este grupo nos encontramos con aquellas aplicaciones, que luego de utilizarlas nos permiten generar prototipos con un nivel bajo de detalle. No suelen ofrecer grandes capacidades para poder personalizar los prototipos. Muchas de estas se basan en componentes predefinidos por la plataforma y en ocasiones no resulta posible modificarlos.

Una de las mejores herramientas para crear prototipos, bocetos o wireframes es **Balsamiq Mockups**[5]**.** Este programa tiene varias ventajas: permite escoger entre una serie de **objetos prediseñados** como barras de estado, menús, barras de

[5] https://balsamiq.com/

progreso, etc. Por un lado, permite **exportar el diseño** en PNG, PDF e incluso al portapapeles, así como incrustar los diseños en una página web o en informes de errores. Además, cuenta con la capacidad de generar prototipos interactivos sobre los cuales es posible navegar entre las distintas pantallas presionando en los botones presentes en cada una de ellas.

Mockflow[6], es otra herramienta que nos permite dibujar interfaces de usuario. Funciona de una forma muy similar a Balsamiq, permitiéndonos crear prototipos a partir de componentes ya predefinidos por la aplicación. Por otro lado, a diferencia de la anterior herramienta, esta última nos permite desarrollar nuestros prototipos utilizando muchos componentes de terceros. De esta manera, si intentamos realizar un prototipo para una plataforma específica, podemos conseguir que nuestra representación use componentes particulares de dicha plataforma. Ofrece integración con diferentes librerías de componentes visuales como Material Design, Bootstrap, iOS 14 entre otros. El funcionamiento es muy similar a Balsamiq, pero ofrece mayor nivel de detalle en los prototipos generados. La interacción que ofrecen a los usuarios estas aplicaciones suele ser muy sencilla y el contenido es bastante estático, sin muchos detalles.

Como conclusión, podemos decir que, si bien este tipo de herramientas pueden llegar a ser muy útiles en las etapas tempranas del desarrollo de un producto de software, los resultados obtenidos no ofrecen grandes diferencias respecto a los prototipos en papel. Su nivel de interacción suele ser muy reducido y los resultados que se consiguen al utilizar estas aplicaciones suelen ser solamente imágenes de los prototipos con ciertas áreas en las cuales es posible presionar e ir a otros wireframes. Son soluciones más profesionales que los prototipos en papel, pero en general ofrecen muchas limitaciones al momento de hacer la representación.

2.1.2 Prototipos de alta fidelidad sin interacción

En esta sección se encuentran aquellas aplicaciones cuyos resultados representan de manera muy cercana los requerimientos de los diseñadores. En ellos podremos observar con un alto nivel de detalle cómo debería verse el resultado final,

[6] https://www.mockflow.com/

casi como si estuviera terminado. Las aplicaciones finales, generadas luego de estos prototipos, no suelen tener grandes diferencias visuales con los diseños planteados por el equipo de diseñadores en los prototipos. Sin embargo, este tipo de prototipos no nos permite comprender el flujo completo de la aplicación ni interactuar con los mismos. Suelen ser representaciones gráficas del diseño de las aplicaciones.

En este grupo podemos encontrar las siguientes aplicaciones:

Sketch: una aplicación de escritorio de diseño vectorial pensada para diseñadores, que permite realizar el trabajo por capas y se especializa en el diseño de interfaces. Ofrece soluciones mucho más fáciles y rápidas que otras herramientas similares, que te permiten crear proyectos interesantes. Trabaja con gráficos vectoriales y cuenta con plantillas, pero no permite ningún tipo de animación más allá de las típicas interacciones para cambiar de página. En resumen, tiene buenos gráficos, pero se queda algo corto en cuanto a interacciones o animaciones.

Para animar en Sketch App existen varios plugins que permitirán hacerlo, pero de manera nativa el programa no tiene esa posibilidad. En el caso de este programa, solamente está disponible para el sistema operativo macOS.

Zeplin: es la herramienta colaborativa para diseñadores y desarrolladores que permite generar componentes y código desde el diseño a medida tanto en iOS, Android y Web. La ventaja de esta herramienta es que al diseñador no le hace falta crear documentos con especificaciones, Zeplin las crea automáticamente, y al desarrollador no le hace falta aprender una herramienta de diseño nueva. Zeplin muestra las medidas, colores, márgenes, imágenes etc., de cada elemento y escribe el código CSS. También Sass y Less entre otros. Además, exporta todas las imágenes e iconos a tantas resoluciones como sea preciso, en un clic.

La mayor ventaja de esta aplicación es que es capaz de generar el código de los estilos en diferentes formatos, facilitando el proceso de desarrollo de la UI. Puede exportar para React Native, Swift y strings generales.

Como pudimos observar, en este grupo se encuentran aquellas aplicaciones que se centran exclusivamente en la representación visual de los prototipos. Su principal objetivo es plasmar los intereses del equipo de diseño para cada una de las pantallas necesarias. Sin embargo, presentan algunas limitaciones ya que ninguna de ellas tiene posibilidad de agregar interacción entre los prototipos. Los resultados obtenidos suelen ser imágenes o bien algunas especificaciones para el equipo de desarrolladores.

Para poder aprovechar estas aplicaciones se requiere la contratación de diseñadores especializados para poder construir el diseño las pantallas.

Este tipo de aplicaciones representan un nivel mucho mayor de detalle que los prototipos en papel. El problema que tiene esto es que, al ser prototipos con tanta información visual, muchas veces la evaluación de usabilidad de estos prototipos suele reducirse a aspectos estéticos y no funcionales. A su vez, al no ofrecer interacción suele resultar más complejo entender el funcionamiento completo del prototipo con todas sus pantallas.

2.1.3 Prototipos de alta fidelidad con interacción

En este grupo nos encontramos con aplicaciones que van un paso más allá de las últimas explicadas en la sección 2.1.2 Estas aplicaciones no solamente nos permiten obtener imágenes, *assets* gráficos, y recursos tales como medidas para poder realizar el desarrollo. Aquí nos encontramos con los prototipos con mayor nivel de detalle existente.

En estas aplicaciones podemos modelar aspectos tales como la interacción entre pantallas, diferente tipo de animaciones, distintos estados dependiendo del contexto, prototipos que contemplan múltiples casos diferentes de navegación entre otras muchas opciones. En este grupo se encuentran:

InVision: una potente **herramienta de prototipado** creada para **especialistas de diseño web** pensada por diseñadores. Una aplicación online que ayuda a generar un diseño espejo de lo que será la web final de manera muy sencilla y ágil para cualquier diseñador online. A la hora de realizar la transición de una página a otra puedes personalizar la animación de cada elemento. No hace falta tener conocimientos de HTML para poder usar esta herramienta.

Adicionalmente, esta aplicación permite la interacción con otros usuarios. Es decir, una o más personas pueden revisar el proyecto de manera online y dejar sus observaciones en el mismo boceto en tiempo real. Esto de cara a la ejecución del proyecto proporciona mucha fluidez y hace ganar tiempo tanto al desarrollador como al cliente. También proporciona la gran diferencia entre enviar al cliente un conjunto de archivos *raster* de todas las pantallas de la futura web, a un link del prototipo web navegable al igual que una web real.

La ventaja que tiene Invision fuera del campo del diseño, es que permite compartir los diseños con el equipo y clientes. Además, permite sincronizar con Sketch App.

Adobe XD o Adobe Experience Design: Es una herramienta de edición de gráficos que funciona para crear interfaces de páginas web y de aplicaciones. Permite al diseñador enfocarse en la experiencia del usuario al navegar, con un rango mínimo de error y en el menor tiempo posible.

Esta herramienta permite diseñar interfaces digitales para aplicaciones móviles, PC, Mac, o sitios web. También permite crear una imagen preliminar de la interfaz de algún proyecto en el que se esté trabajando de manera más sencilla que con otras herramientas.

Una de las ventajas es que ayuda a ahorrar tiempo, ya que la imagen preliminar da la oportunidad de hacer cambios rápidos, en caso de ser necesarios, antes de llegar a la etapa de programación. Permite crear prototipos interactivos que simulan la navegación real en el proyecto.

En comparación con el grupo anterior, podemos ver cómo algunas de las limitaciones de dicho grupo fueron solucionadas. Este grupo agrega la posibilidad de interacción con los prototipos, pero se sigue necesitando de diseñadores para poder armar los diseños. A su vez, continúan centrando la mirada en aspectos visuales, aunque mejoran la percepción de algunos aspectos funcionales al proveer mayor nivel de interacción en los prototipos generados.

En general, al utilizar este tipo de aplicaciones, el próximo paso suele ser el desarrollo de las aplicaciones para las cuales se hicieron los prototipos. Si bien ofrecen excelentes niveles de detalle, el gasto asociado a la creación de estos prototipos suele ser mucho mayor que con las aplicaciones anteriores. El costo de contratar un equipo de diseño para armar el prototipo suele ser considerable.

A su vez, si queremos hacer alguna prueba sobre el producto generado solamente podremos hacerlo compartiendo el prototipo generado, donde la experiencia de probar el prototipo suele estar guiada por las diferentes plataformas. No nos permiten realizar pruebas de los prototipos de forma libre, como lo haría un usuario real con el software desarrollado.

Si bien la utilización de estas aplicaciones resulta de mucha utilidad, en general no nos permiten realizar pruebas de usabilidad de forma temprana. Este tipo de aplicaciones están pensadas para poder desarrollar los prototipos con los mayores niveles de detalle posible. No contemplan la posibilidad de poder hacer pruebas sobre distintas versiones de una misma aplicación.

2.1.4 Herramientas que nos permiten generar el código de las aplicaciones basadas en prototipos.

Estas aplicaciones utilizan una combinación de técnicas de visión por computadora para clasificar las formas dibujadas en una superficie digital en componentes predefinidos. En el artículo ***Mobidev**: a tool for creating apps on mobile phones* (Seifert, J. et al. 2011) se describe la aplicación móvil desarrollada, mientras que en el paper ***Sketching interfaces**: toward more human interface design* (Landay, J. A., & Myers, B. A., 2001) se describe una aplicación desktop.

Ambas aplicaciones nos han demostrado que, al utilizar técnicas de visión por computadora como la detección de contornos, la detección de bordes o márgenes y la detección de "líneas", permite clasificar de manera sólida las formas de elementos y componentes de la aplicación, como botones y texto.

La segunda área que analizamos trata de la traducción de prototipos de alta fidelidad en código como se describe en el artículo *Reverse engineering mobile application user interfaces with remaui (t).* (Nguyen, T. A., & Callner, C., 2015) **REMAUI**, que está diseñada para traducir un prototipo realizado en una aplicación o incluso una captura de pantalla de una página web existente en código. Este tipo de herramientas no solo tomarán la estructura brindada, sino que también los estilos y las fuentes.

REMAUI utiliza técnicas similares a MobiDev para identificar y clasificar elementos de la imagen como: imágenes, texto, título, botones, menús, entradas, etc.

Otra aplicación que convierte diseños de alta fidelidad en código es **pix2code**, Figura 3, con simplemente una captura de pantalla es suficiente para aplicar deep learning y predecir el código. Así como describe Beltramelli en *pix2code: Generating code from a graphical user interface screenshot. (Beltramelli, T. 2018)* aunque esta aplicación tenga grandes ventajas sobre las aplicaciones previamente mencionadas, no es capaz de generar un ejemplo del mundo real.

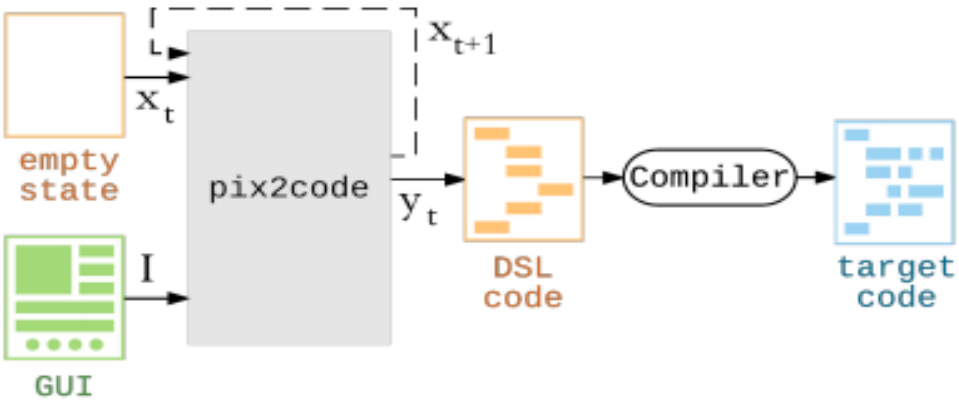

Figura 3 Descripción del training network de pix2code, ejemplo completo disponible en el paper.

Sketch2Code[7]**:** es una aplicación web que convierte cualquier dibujo a mano alzada de una página web en su correspondiente código HTML. El código generado es posible descargarlo y utilizarlo. Esta aplicación es un proyecto de Microsoft y trabaja con datasets que contienen 150 componentes de ejemplo, lo que para diferentes aplicaciones puede llegar a ser un número acotado.

Luego de leer los artículos nombrados e investigar las aplicaciones existentes, nos encontramos con algunos potenciales problemas:

- En general, todas estas aplicaciones están centradas en digitalizar una única pantalla y no ofrecen forma de poder desarrollar un prototipo completo de una aplicación real. Cada wireframe es digitalizado de forma individual y si queremos armar una aplicación completa con ellos será necesario buscar la forma de poder combinar cada una de las interfaces generadas.

- Algunas de estas aplicaciones, si bien nos permiten convertir imágenes en código, muchas veces requieren de prototipos sumamente detallados y obtenidos de aplicaciones reales para poder armar el código. Este es el caso de **REMAUI** o **Pix2Code** que para armar el prototipo lo hacen en base a un producto real ya creado y esto no nos permitiría hacer nuestros

[7] https://sketch2code.azurewebsites.net/

propios diseños personalizados. Siempre necesitamos contar con alguna aplicación real o prototipo previamente realizado para poder hacer la conversión a código.

- Ninguna de estas aplicaciones nos permite poder realizar pruebas de usabilidad directamente sobre los prototipos generados.

- Al hacer uso de técnicas de Deep Learning o Machine Learning, las aplicaciones requieren mucho tiempo para ampliarse en caso de que se necesite introducir nuevos elementos, ya que dependen de los programadores para diseñar características que coincidan con los dibujos que un usuario puede ingresar.

- La falta de atención en el pos-procesamiento de los componentes detectados para corregir errores en el prototipo. Por ejemplo, un usuario puede no dibujar dos elementos perfectamente alineados cuando lo deseaba, esto da como resultado una representación no acertada del wireframe donde los elementos se colocan exactamente dónde están.

- La falta de atención en el pos-procesamiento de los componentes detectados para corregir errores en el prototipo.

2.1.5 Conclusiones sobre las aplicaciones relacionadas

Luego de la descripción de cada una de las herramientas que hoy en día se encuentran disponibles para la construcción y digitalización de prototipos, logramos llegar a la conclusión que cada una de estas aplicaciones fueron pensadas, diseñadas y creadas para aportar al trabajo de los diseñadores y desarrolladores con una funcionalidad específica.

En el caso de Zeplin e InVision, se destacan fuertemente en la parte técnica de una aplicación, ya que a partir de un prototipo digital se enfocan en ofrecer información de utilidad a la hora de generar código, usando de base el diseño cargado por los diseñadores. Se pueden generar diseños tanto para iOS, android y web, además permiten agregar navegación entre los diferentes prototipos creados.

Por otro lado, están las aplicaciones como Sketch, Balsamiq y Adobe XD, donde no podremos generar ayudas de código ya que su principal funcionalidad está dada por

la creación estática de mockups y prototipos, los cuales se podrán exportar como PDF, imágenes, etc. Brindan una serie de componentes prediseñados para reutilizar a medida que generamos nuevos prototipos estáticos.

Nuestra aplicación fue pensada estratégicamente para que incluya las mejores características de cada una de las aplicaciones ya existentes en el mercado e incluso añadimos nuevas características que marcan la diferencia con el resto de las aplicaciones. De la aplicación de manipulación de prototipos en papel que desarrollamos para esta tesina, se destacan las funcionalidades principales como:

1. Digitalización sencilla de los prototipos en papel.
2. Interacción real de los usuarios con los prototipos creados. Cada elemento, botón, select, etc. permite ser utilizado.
3. Desarrollo de pruebas de usabilidad para comprender mejor cuán probable es que los usuarios puedan completar diferentes tareas, permitiendo a los diseñadores, dueños de producto y desarrollo validar con posibles usuarios finales el prototipo creado.
4. Elaboración de un conjunto de métricas de usabilidad que brinden información empírica sobre cómo los usuarios interactúan con nuestro prototipo.

Además, MWP se diferencia por la facilidad con que los usuarios podrán realizar el proceso de digitalización de las diferentes vistas, gracias al desarrollo que realizamos para que sea un paso a paso guiado hasta llegar a la previsualización de la vista que se desea digitalizar.

Al ser una aplicación móvil, basta con descargar la aplicación de la App Store para poder llevarla con nosotros a todas las reuniones que tengamos con los clientes donde en simples pasos y en muy pocos minutos podremos obtener un prototipo digitalizado navegable para debatir y llegar a un acuerdo entre la idea que el cliente tenga en mente y los requisitos solicitados.

Permitir la generación de un prototipo casi inmediatamente acortará los tiempos de la elicitación de requerimientos y le brindará mayor confianza al cliente, ya que puede visualizar exactamente cómo funcionará su aplicación una vez que la etapa de implementación culmine.

También beneficiará tanto a los diseñadores como desarrolladores para mejorar aspectos de usabilidad y diseño de experiencia de usuario de la aplicación que se

encuentra en etapa de prototipado de manera prematura, abordando con cautela dichos aspectos podremos desarrollar una aplicación que sea útil para todas las personas. Poder realizar pruebas empíricas de usabilidad y obtener métricas de las mismas nos ayuda a entender mejor el comportamiento de los usuarios, sin necesidad de tener la aplicación terminada para que podamos obtener esta información.

2.2 Trabajos sobre usabilidad

La obtención de requisitos es un proceso difícil en el que uno tiene que lidiar con la ambigüedad, la informalidad, la incompletitud y la inconsistencia, en el que el "conocimiento" de los requisitos no es claro. Es un aspecto importante y fundamental en el desarrollo de software.

La creación de prototipos es una de las técnicas utilizadas durante la etapa de ingeniería de requisitos, en lugar de prototipos costosos, *se sugiere un método de prototipo de papel desechable*. Puede dibujarse a mano o crearse mediante un programa de gráficos.

Por lo general, el prototipo de papel se utiliza como parte de las pruebas de usabilidad, donde el usuario se familiariza con la interfaz de usuario, como describe Vijayan J. y Raju G. en el paper *A new approach to requirements elicitation using paper prototype (Vijayan J. & Raju G., 2011)*.

Independientemente de las técnicas utilizadas, los conceptos básicos siguen siendo los mismos. La ingeniería de requisitos se describe en 5 pasos: *Solicitud de requisitos*, *Análisis de requisitos*, *Especificación de requerimiento*, *Validación de requisitos* y *Gestión de requerimientos*. Una vez finalizado el prototipo se estabilizan los requisitos. La especificación de requisitos se realiza después del proceso de estabilización y el prototipo se puede descartar.

Aunque no son componentes de la definición de ISO, diferentes profesionales (Gould y Lewis 1985; Shackel, 1990; 1991; Sharp, Rogers y Preece 2007; Stone et al. 2005) han considerado durante mucho tiempo los siguientes aspectos como parte de la usabilidad:

- **Flexibilidad**: la medida en que el sistema puede adaptarse a los cambios deseados por el usuario más allá de los especificados en primer lugar.

- **Capacidad de aprendizaje**: el tiempo y el esfuerzo necesarios para alcanzar un nivel específico de rendimiento de uso con el sistema (también conocido como facilidad de aprendizaje).

- **Memorabilidad:** el tiempo y el esfuerzo necesarios para volver a un nivel específico de rendimiento de uso después de un período específico fuera del sistema.

- **Seguridad:** aspectos del sistema relacionados con la protección del usuario de condiciones peligrosas y situaciones indeseables.

El planteo anterior muestra que la usabilidad no tiene una definición absoluta, sino que está relacionada con los usuarios, los objetivos y los contextos de uso que son apropiados para el conjunto particular de circunstancias (Petrie, H., & Bevan, N.,2009).

Por ejemplo, si uno está desarrollando un sistema de reserva online para aerolíneas, para que los agentes de viajes profesionales lo utilicen en el trabajo, los requisitos o criterios para los componentes de usabilidad, como la eficiencia y la capacidad de aprendizaje, sin duda serán diferentes que si uno está desarrollando un sitio web para que el público en general reserve pasajes de avión.

Dado que en la usabilidad abordamos las posibilidades de acción que un usuario percibe para realizar una tarea frente a "algo" -como una interfaz de usuario GUI, por ejemplo- y que estas opciones les resultan más o menos fáciles según sus propios modelos mentales (las experiencias anteriores que han vivido), podríamos decir que es un fenómeno muy cercano al *affordance* en el mundo material.

A partir de aquello, utilizar métricas de usabilidad nos ayudaría a medir de forma empírica la usabilidad, con muestras tomadas a partir de usuarios reales, en escenarios donde deben completar ciertas tareas que serán capturadas para evaluación y el posterior análisis de los resultados cuantitativos y cualitativos en términos de *eficacia*, *eficiencia* y *satisfacción*; incluso, nos permitiría conocer su grado de accesibilidad.

- ***Efectividad***: relacionado con la precisión y completitud con la que los usuarios utilizan la aplicación para alcanzar objetivos específicos. La calidad de la solución y la tasa de errores son indicadores de efectividad.

- ***Eficiencia***: relación entre efectividad y el esfuerzo o los recursos empleados para lograr una determinada tarea. Indicadores de eficiencia incluyen el tiempo de finalización de las tareas y el tiempo de aprendizaje. A menor cantidad de esfuerzo o recursos, mayor eficiencia.

- ***Satisfacción***: grado con que el usuario se siente satisfecho, con actitudes positivas, al utilizar la aplicación para alcanzar objetivos específicos. La satisfacción es un atributo subjetivo, puede ser medido utilizando escalas de calificación de actitud. Para poder especificar o medir la usabilidad, es necesario descomponer los atributos y el contexto de uso en componentes medibles y verificables. Las relaciones que existen entre el usuario, el producto, los atributos, el contexto de uso y los objetivos que se quieren lograr se pueden observar en el framework de usabilidad propuesto en la norma citada.

En el año 2000 *Jakob Nielsen* y *Tom Landauer* postularon en el artículo *"Why You Only Need to Test With 5 Users"* (Nielsen, J., 2000) que, matemáticamente, con un test de 5 usuarios es posible descubrir el 90% de los problemas al realizar una prueba de usabilidad. Este modelo ha sido validado en diversos productos de Google donde se han realizado pruebas de usabilidad y entrevistas en profundidad a no más de 5 usuarios, con excelentes resultados.

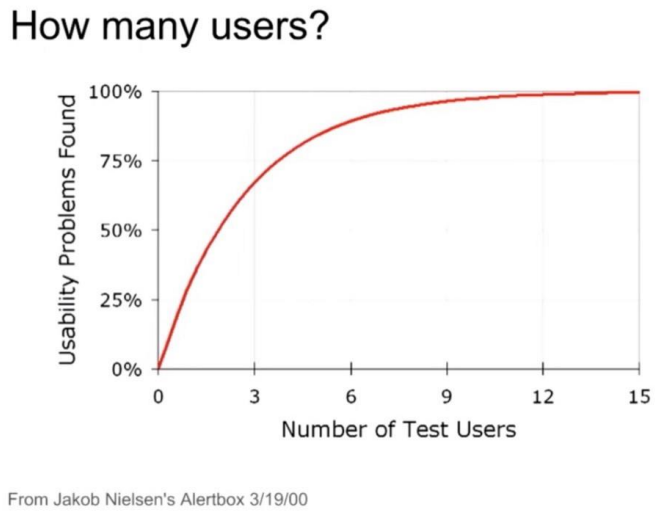

Figura 4 Evaluación de los resultados de las pruebas de usabilidad con 5 usuarios.

Por lo tanto, es importante tener en claro que este test no nos dirá qué tipo de acciones debemos realizar para resolver los problemas puntuales de diseño, pero sí qué aspectos problemáticos debemos tener en cuenta para presentar soluciones que atiendan sus expectativas.

Diferentes percepciones de usabilidad resultan en una variedad de estructuras de medidas, estas estructuras definen los atributos y sus métricas asociadas. Por ejemplo, Jakob Nielsen, define usabilidad en términos de cinco atributos: facilidad de aprendizaje, eficiencia, memorabilidad, errores y satisfacción.

A la vez, los atributos de usabilidad, pueden ser clasificados en objetivos y subjetivos. Los atributos objetivos pueden ser medidos a través de la interacción del usuario con la aplicación, no dependen de la percepción del usuario; en cambio los subjetivos están relacionados con el factor humano, se refiere a la actitud del usuario hacia el uso de la aplicación, está vinculado a las emociones y por lo tanto son más difíciles de medir y cuantificar.

Debido a que los atributos de una aplicación son conceptos abstractos, estos no pueden ser directamente medidos. Para medirlos se les asocian distintas métricas, por ejemplo, el atributo eficiencia puede ser evaluado mediante la métrica que calcula el tiempo empleado por un usuario en terminar una tarea específica.

Una métrica debe cumplir con ciertas características:

- Debe ser posible de medir.
- Tiene que poseer un rango determinado de valores que puede tomar, los cuales deben estar previamente establecidos al momento que se define.
- Necesita tener un valor mínimo y máximo deseable, los cuales sirvan de referencia para poder evaluar cuan cerca o lejos se está del objetivo esperado.
- Cuando una métrica representa una característica que aumenta cuando se presentan rasgos positivos o que disminuye al encontrar rasgos indeseables, el valor de la métrica debe aumentar o disminuir en el mismo sentido.
- Cada métrica debe validarse empíricamente en una amplia variedad de contextos antes de publicarse o aplicarse en la toma de decisiones.

Cabe aclarar que las métricas no representan un fin por sí mismas, estas revelan datos e información sobre la experiencia personal del usuario cuando hace uso de una aplicación. La información obtenida de las métricas nos ayuda a realizar un mejor

análisis y tomar decisiones más acertadas con respecto a la usabilidad de una aplicación.

La siguiente Tabla 1, muestra atributos de usabilidad y las métricas comúnmente asociadas a los mismos para poder cuantificarlos, también descrito por *DeMarco* en el paper *Controlling software projects: Management, measurement, and estimates* (DeMarco, T., 1986).

Atributos	**Métricas**
Efectividad	● Tareas resueltas en un tiempo limitado. ● Porcentaje de tareas completadas con éxito al primer intento. ● Número de funciones aprendidas.
Eficiencia	● Tiempo empleado en completar una tarea. ● Número de teclas presionadas por tarea. ● Tiempo transcurrido en cada pantalla. ● Eficiencia relativa en comparación con un usuario experto. ● Tiempo productivo.
Satisfacción	● Nivel de dificultad. ● Agrada o no agrada. ● Preferencias.
Facilidad de aprendizaje	● Tiempo usado para terminar una tarea la primera vez. ● Cantidad de entrenamiento. ● Curva de aprendizaje.
Memorabilidad	● Número de pasos, clicks o páginas usadas para terminar una tarea después de no usar la aplicación por un periodo de tiempo.

Tabla 1 Atributos de usabilidad y las métricas comúnmente asociadas a los mismos.

Dependiendo de la naturaleza de la aplicación a estimar, en las pruebas se da relevancia a diferentes atributos. La usabilidad del sistema no es una simple adición del valor de estos atributos, sino que se define para cada sistema como un nivel a alcanzar por cada uno de los atributos especificados para ese sistema.

La evaluación de la usabilidad de una aplicación de software, consiste en realizar pruebas para obtener medidas e información y observar debilidades relacionadas al uso de la misma.

Como menciona *DeMarco* en el paper *Controlling software projects: Management, measurement, and estimates* (DeMarco, T., 1986), las cuatro formas básicas de evaluación son: automática (se calculan las métricas mediante la ejecución de la aplicación), empírica (la usabilidad es evaluada testeando la aplicación con usuarios reales), formal (usando modelos formales y fórmulas para el cálculo de medidas de usabilidad), e informal (basados en reglas generales y la habilidad y experiencia de los evaluadores).

Capítulo 3: Un proceso ágil basado en prototipos de papel digitalizados

En esta sección introduciremos el proceso general para la evaluación de usabilidad de manera temprana. Este proceso está dividido en diferentes pasos. Comienza con la construcción de los wireframes, que serán digitalizados en la aplicación MWP. Posterior a eso, se realizarán pruebas de usabilidad con usuarios para obtener métricas que nos permitan comprender como los usuarios interactúan con los prototipos digitalizados.

A través de la implementación de la metodología se buscará brindar soluciones a diferentes problemáticas relacionadas con el diseño, usabilidad y a la elicitación de requerimientos.

Metodología para evaluar usabilidad de manera temprana

En esta sección se describen las distintas etapas que conforman la metodología para la evaluación de usabilidad de manera temprana, Figura 5. La primera etapa consiste en plasmar las ideas en papel y dibujar los diferentes wireframes que comprenderán la aplicación a construir.

Una vez definidos los wireframes, continuaremos con la digitalización de éstos en la aplicación para manipulación digital de prototipos. En este paso, haremos la carga y procesamiento de los wireframes dibujados en el primer paso para poder obtener una previsualización de los prototipos digitalizados.

En este punto es donde se selecciona cada posible widget existente en cada pantalla, y se asignan los atributos correspondientes a los mismos. De esta forma podremos personalizar la representación de acuerdo a diferentes propiedades que pueden establecerse en cada widget según el tipo de elemento que queramos representar.

Siguiendo con esta metodología deberemos plantear cuáles son los casos de uso que deseamos evaluar y qué aspectos debemos considerar al momento de analizar los resultados de las pruebas.

Habiendo concluido la etapa de la digitalización, y teniendo los objetivos de las posibles pruebas de usabilidad, crearemos y realizaremos las mismas con usuarios para obtener resultados concretos sobre el funcionamiento, completitud y usabilidad de los prototipos.

Una vez finalizadas las pruebas con los usuarios, podremos obtener diferentes métricas que nos permitan evaluar si los prototipos cumplieron o no los objetivos esperados. De acuerdo a los resultados obtenidos podremos determinar si los prototipos creados son capaces de responder a las necesidades del negocio. A su vez, también conoceremos con la opinión de los usuarios que participaron de las pruebas.

En esta instancia, luego de haber analizado las métricas y los comentarios recibidos se podrá decidir cómo continuar con el proceso. En este punto existen 2 posibles caminos a seguir: si los resultados de las pruebas fueron favorables se continuará con la implementación de los prototipos evaluados, caso contrario, es

posible realizar cambios en los prototipos y ejecutar nuevas pruebas de usabilidad para comparar los resultados obtenidos.

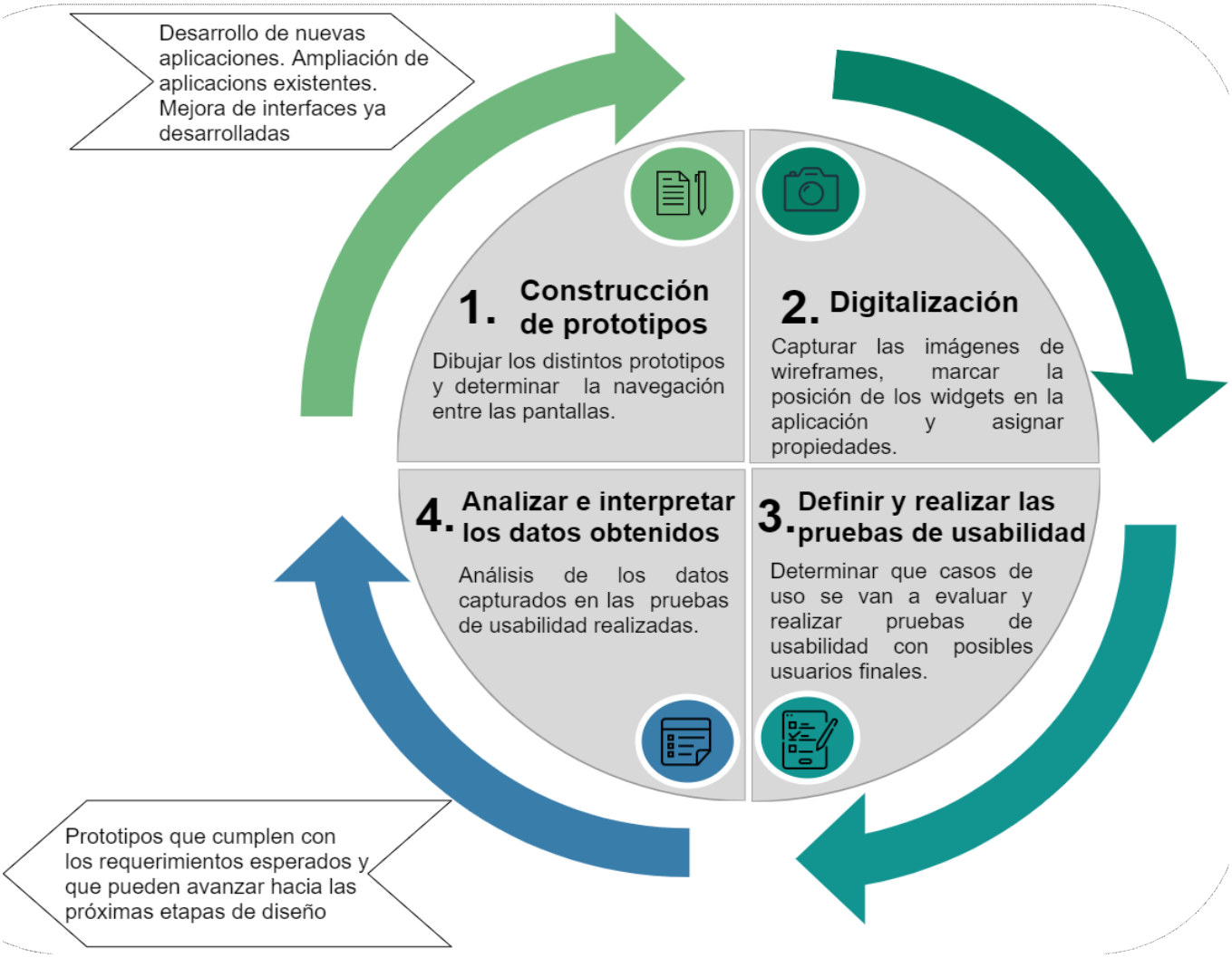

Figura 5 Flujo de la metodología llevada a cabo en la presente tesina.

3.1 Construcción de los prototipos en papel

En estas etapas iniciales con los prototipos en papel es cuando se realiza la evaluación de los requerimientos. A partir de los prototipos en papel obtenidos podemos establecer correcciones o posibles alternativas para cada una de las pantallas o flujos a desarrollar. Por otra parte, también nos permiten comprender la totalidad de

una aplicación mediante la combinación de diferentes wireframes que pertenezcan al mismo proyecto.

Para la realización de los prototipos en papel, se debe considerar la cantidad de pantallas necesarias, así como también el respectivo análisis de cada una de ellas. Por lo general, cada pantalla tendrá su propio prototipo en papel y este a su vez contendrá diferentes widgets que lo compongan. Además, mediante la utilización de prototipos en papel podemos obtener rápidamente diferentes versiones de un mismo proyecto o pantalla y a su vez ir comparando los resultados entre ellos.

Luego de completada la elicitación de requerimientos, y como resultado de esta primera etapa, obtendremos un conjunto de wireframes que representan una aplicación completa o alguna determinada funcionalidad que se desea realizar.

En los próximos hitos del proceso, veremos cómo mediante la digitalización de los prototipos podremos evaluar la usabilidad de los mismos en pruebas con posibles usuarios finales. Dichas pruebas nos permitirán conocer sus niveles de satisfacción y comprender mejor cómo los usuarios se comportan frente al prototipo digitalizado de nuestra aplicación. De esta manera, mediante la validación de los prototipos con usuarios, podremos detectar problemas de usabilidad que afecten negativamente a la interacción que los usuarios tengan con el producto.

3.2 Digitalización de los prototipos en papel creados

Una de las principales desventajas de los prototipos en papel es que no se puede generar interacción de ninguna manera con el usuario. Por esta razón, es necesario para continuar con la metodología realizar la digitalización de los mismos.

Mediante la digitalización de los prototipos, podremos acercar a los usuarios a una versión interactiva del producto que puedan utilizar, y así, de esta manera observar cómo utilizan la aplicación e interactúan con los diferentes elementos visuales que compongan el prototipo digitalizado.

Además de brindarle interacción a los usuarios, lograríamos evitar situaciones inesperadas al destacar los requerimientos incompletos, inconsistentes o la falta de determinadas funcionalidades no determinadas en las etapas de elicitación de requerimientos. Por otro lado, también se verán reducidos los costos de rediseño si los

problemas se detectan pronto y son fáciles de identificar previniendo posibles problemas futuros de usabilidad o requerimientos.

3.3 Definir y realizar las pruebas de usabilidad con usuarios

Para poder medir la usabilidad del prototipo de una aplicación es importante saber qué aspectos se quieren medir y entender cómo es que se pueden realizar las distintas mediciones correspondientes. Para ello es recomendable analizar diferentes tareas puntuales y en función de dichas tareas y resultados tomar las medidas necesarias para poder mejorar la usabilidad en cada tipo de funcionalidad.

Es importante en este punto determinar cuáles son los casos de prueba que se desean validar. Una aplicación puede responder a múltiples flujos y tareas. La cantidad completa de casos de uso disponibles para un mismo desarrollo suele ser muy extensa. Medir una nueva aplicación de manera completa, sin dividirla en diferentes casos de uso puede resultar muy complejo y difícil de entender.

Para esto se sugiere determinar los diferentes casos de prueba para los cuales se desee ejecutar las pruebas de usabilidad. Es recomendable analizar diferentes tareas puntuales y en función de dichas tareas y resultados tomar las medidas necesarias para poder mejorar la usabilidad en cada tipo de funcionalidad.

Previo a realizar las pruebas, debemos conocer con exactitud qué necesitamos medir y analizar. En función de eso se podrán determinar objetivos claros los cuales busquemos alcanzar. Con esos objetivos en mente, se deberán definir los valores de referencia para las distintas métricas y atributos a evaluar.

En función de cada caso de uso que se desea analizar se podrán crear diferentes pruebas. A partir de estas pruebas, se obtendrán las métricas de usabilidad según los valores definidos, mediante los cuales se podrá medir la usabilidad de nuestros prototipos tomando como parámetro los valores de referencia que se hayan establecido para cada métrica que se necesite evaluar.

A partir de estos objetivos, métricas y valores de referencia podremos determinar si los prototipos evaluados cumplen o no con los objetivos planteados al momento de su construcción. Luego de haber definido los prototipos a evaluar junto

con sus casos de prueba y concluida la digitalización de los mismos, se podrán realizar las pruebas de usabilidad.

En esta etapa es cuando los posibles usuarios finales utilizan los prototipos digitalizados y dan sus conclusiones en cuanto a las pruebas que realizan, el funcionamiento de los prototipos y el nivel de facilidad para utilizarlos.

El hecho de realizar pruebas de usabilidad de manera temprana con usuarios nos permite evitar futuros errores y poder brindar una solución acorde. Durante las pruebas de usabilidad, la respuesta de los usuarios es de suma importancia, ya que nos dará la pauta de que tan contentos están con el prototipo y dependiendo de los resultados obtenidos se podrá estimar el nivel de éxito que podrían alcanzar los mismos.

3.4 Interpretación de las métricas de usabilidad obtenidas en las pruebas con usuarios finales

En este punto ya habremos realizado diferentes pruebas de usabilidad de los prototipos con múltiples usuarios. Tal como observamos en la sección 2.2 , existen diferentes atributos para medir la usabilidad de una aplicación. Algunos de ellos son: efectividad, eficiencia, satisfacción, facilidad de aprendizaje o memorabilidad. A su vez, dentro de cada uno de ellos existen diferentes métricas que nos permiten evaluarlos.

Para medir la **Efectividad** podemos evaluar qué porcentaje de las pruebas finalizó correctamente habiendo completado el objetivo planteado sobre el total de pruebas realizadas.

La **Eficiencia** puede evaluarse con múltiples métricas diferentes. Dentro de las pruebas podemos medir el tiempo total para completar la misma como así también el tiempo transcurrido en cada pantalla. A su vez, para cada una de las diferentes ejecuciones de las pruebas podemos saber si el usuario participante es la primera vez que realiza la prueba o ya ha participado de la misma anteriormente.

Con los parámetros anteriores podremos evaluar la eficiencia en base al tiempo total empleado para realizar la prueba o el tiempo total en cada pantalla. De esta forma, si realizamos cambios en los prototipos que estamos evaluando, podremos determinar si con los cambios efectuados la eficiencia aumenta o disminuye de acuerdo al tiempo empleado por los usuarios para completar la misma tarea.

Si quisiéramos medir **Satisfacción**, podremos hacerlo mediante la calificación que los usuarios emiten sobre el prototipo al finalizar la prueba. Los voluntarios de las pruebas podrían valorar cuán sencillo fue alcanzar el objetivo planteado con el prototipo o qué nivel de satisfacción tuvieron con la prueba realizada.

Evaluar la **Facilidad de Aprendizaje** puede determinarse conociendo el tiempo que los usuarios tardan para completar alguna prueba de usabilidad. Para cada prueba realizada se mide el tiempo total empleado para realizar la misma. A mayor sea el tiempo obtenido más difícil será de aprender la tarea que se esté evaluando.

A mayor sea la efectividad, facilidad de uso, o eficiencia, mejor será la usabilidad del prototipo creado. Sin embargo, con esto solo no nos alcanza para estar seguros de que el prototipo es correcto de acuerdo a las necesidades de negocio.

Es importante asegurarse que los prototipos cumplen los requerimientos mínimos esperados por los clientes. Esto se podrá verificar luego de digitalizados los prototipos con las personas interesadas en el desarrollo del mismo, validando que sean correctos de acuerdo a sus necesidades y estén completos.

Por otro lado, para asegurarnos que nuestro prototipo es válido y puede avanzar en las próximas etapas del desarrollo del producto, tenemos que evaluar y comparar los valores obtenidos en las métricas con los valores esperados que se consideren aceptables.

En caso de que los valores alcancen los mínimos requeridos se podrá avanzar con la construcción de los prototipos en productos reales. Caso contrario, será necesario analizar y entender en profundidad cuáles aspectos son necesario mejorar antes de poder avanzar con la construcción del producto.

Suponiendo que los prototipos no cumplen con los resultados esperados se deberán realizar cambios y volver a realizar las pruebas de usabilidad para poder determinar si los cambios aplicados impactaron positivamente en el cumplimiento de los objetivos y si esta mejora es suficiente o se requiere continuar realizando nuevos cambios de diseño para mejorar así la usabilidad de los prototipos.

Cabe destacar que esta metodología puede ejecutarse múltiples veces con diferentes prototipos. De esta forma podemos intentar mejorar los prototipos desarrollados o bien evaluar diferentes alternativas para determinar cuál de ellas nos permite alcanzar los objetivos esperados de la mejor forma posible.

Capítulo 4: Aplicación móvil para digitalización de prototipos en papel y evaluación de usabilidad

En el capítulo 3 se describió una metodología para poder realizar evaluaciones de usabilidad de manera temprana. Para ello establecimos que era necesario realizar un proceso de digitalización de los prototipos en papel que nos permitiera poder obtener métricas de usabilidad de manera empírica. Mediante el análisis de dichas métricas se puede obtener información de las evaluaciones de usabilidad de los prototipos.

Para poder digitalizar los prototipos se necesita de una herramienta que nos permita hacer el proceso de digitalización y posteriormente la ejecución de las diferentes pruebas de usabilidad. Tal como analizamos previamente, no existen herramientas que nos permitan realizar este proceso. Por este motivo, optamos por desarrollar una aplicación móvil que nos permitiera realizar este proceso.

En las siguientes secciones se brindarán mayores detalles sobre la herramienta de manipulación de prototipos creada. Se detallarán los aspectos funcionales y visuales de la aplicación para digitalización de wireframes en papel.

Buscamos expresar en profundidad las alternativas que brinda la herramienta. Veremos cómo se pueden crear nuevos proyectos, cómo se realiza la digitalización de los prototipos en papel y cómo se realizan pruebas de usabilidad que nos permiten conocer muchos aspectos de la usabilidad de los prototipos creados.

4.1 Digitalización de los prototipos en papel

En el proceso de digitalización comenzaremos agregando un nuevo wireframe, se desplegará un modal para completar con el nombre que le asignaremos al wireframe, la sección a la que estará asociado y tendremos también un botón disponible para subir la imagen correspondiente al wireframe. Al igual que funciona en otras aplicaciones, existe la opción de tomar una foto en el momento, elegir una foto existente o seleccionar una foto desde la galería.

Una vez que tengamos la foto cargada en nuestra aplicación, continuaremos el proceso de digitalización, recortando la imagen y realizando los ajustes que sean necesarios. Continuaremos con el cuarto paso del proceso, en el cual tendremos que trabajar con la imagen, seleccionaremos las partes de la foto del wireframe que vamos a representar y guardar posteriormente en la aplicación, para dicho fin utilizaremos reglas verticales y horizontales. Las reglas podrán ser ubicadas a lo largo de toda la imagen.

Una vez que posicionamos las reglas en el lugar deseado, presionaremos, por unos pocos segundos, en la pantalla dentro de la sección que delimitamos con las reglas para que nos reconozca la porción de la imagen donde se digitalizará un componente en el siguiente y último paso.

Como último paso del proceso de digitalización debemos elegir y seleccionar de una lista de componentes, que tipo de componente vamos a digitalizar. Además, luego que seleccionemos el tipo de componente le aplicaremos las características o propiedades que correspondan. Por ejemplo, si lo que estamos digitalizando es un botón, le asignaremos un nombre, podremos elegir un color determinado para el botón, el tamaño, entre otras propiedades más.

El proceso de digitalización finaliza con la confirmación del componente que editamos en los pasos anteriores y que podrá ser visible antes de confirmarse.

4.2 Módulos funcionales de la aplicación

Para poder realizar la digitalización de los prototipos, es importante comprender las diferentes funcionalidades de la aplicación. Para ello hemos decidido

dividirla en cuatro grandes módulos que contengan todas las funcionalidades de la aplicación.

Un módulo para la *Creación de proyectos y wireframes*, el módulo que corresponde a la *Captura y procesamiento de imágenes*, el de *Digitalización de los wireframes* y por último el de *previsualización y pruebas de usabilidad*.

4.2.1 Creación de proyectos y wireframes

Cuando buscamos digitalizar nuestros prototipos, es necesario que podamos crear diferentes proyectos, secciones y wireframes. Mediante el uso de estos elementos podremos pasar de un prototipo en papel a un prototipo digitalizado permitiendo así conseguir una previsualización del mismo o ejecutar diferentes pruebas de usabilidad. El objetivo de este módulo es gestionar todo esto para permitirnos crear un proyecto en su totalidad junto con sus diferentes pantallas.

La siguiente Figura 6, muestra una presentación de la interfaz propuesta en la aplicación de MWP (Manipulación de wireframes en papel). Para facilitar la identificación de las tareas se ha agregado, en el margen inferior, las páginas principales con las que cuenta la aplicación.

Si observamos el botón color azul en el inferior de la imagen podremos **crear nuevos proyectos**. Dentro de los mismos tendremos la posibilidad de ir agregando nuevos wireframes y secciones. Cada wireframe representará una pantalla diferente dentro de la aplicación.

Figura 6 Página principal de la aplicación

Luego de hacer click en el botón de crear proyecto, nos abrirá la siguiente pantalla de la Figura 7, en donde podremos asignarle un nombre y descripción al proyecto.

Figura 7 Imagen que muestra la creación de un proyecto nuevo dentro de la aplicación.

Posterior a la creación del nuevo proyecto, procederemos a agregar nuevos wireframes para digitalizar como se puede observar en la Figura 8.

Figura 8 Creación de un nuevo wireframe dentro de un proyecto.

4.2.2 Captura y procesamiento de imágenes

En este módulo obtendremos las diferentes imágenes de los prototipos en papel que necesitamos para poder representar nuestro proyecto. Estas pueden ser

capturadas desde la aplicación o también podremos hacer uso de las imágenes que actualmente tenemos en nuestra galería, como se muestra en la siguiente Figura 9.

Figura 9 Diferentes opciones para la carga de una imagen.

Como parte del procesamiento de las imágenes, podremos cortarlas en diferentes formatos como, por ejemplo: 1:1; 16:9; 4:3; etc.

4.2.3 Digitalización de los wireframes

Para poder digitalizar los prototipos en papel, es necesario que podamos etiquetar las diferentes partes de la imagen como pueden ser los botones, textos, imágenes, entre otros elementos que pueden ser utilizados en una interfaz de usuario, Figura 10.

Para ello, haremos uso de un sistema de guías verticales y horizontales mediante las cuales podremos seleccionar cada uno de los widgets que vamos a digitalizar. Las guías se extienden desde el comienzo de la imagen, sin importar qué tamaño tenga, hasta el final.

Una vez que tengamos los widgets marcados con las guías podremos seleccionar cada fragmento de la imagen, el cual es identificado con un cuadrado de color para representar las diferentes partes de la imagen donde deberemos situar cada uno de los componentes al momento de representar el wireframe en la previsualización o en las pruebas de usabilidad.

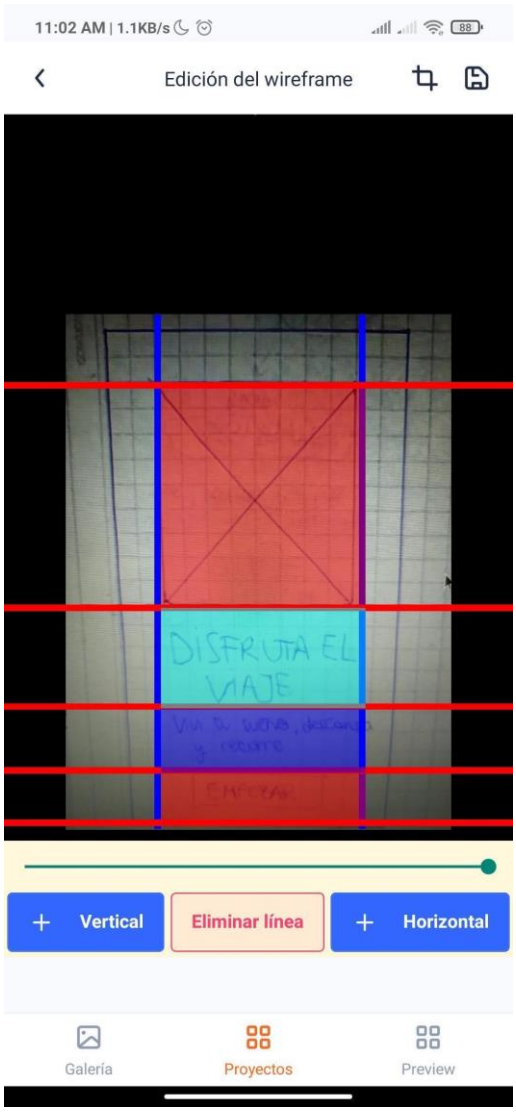

Figura 10 Ejemplo de un prototipo con diferentes componentes etiquetados. Cada uno de los cuadrados de colores representa a un componente diferente dentro de la pantalla.

Continuando con la digitalización de cada una de las pantallas, y luego de haber marcado los diferentes widgets en el sistema de guías podremos etiquetar cada componente. Haremos esto para decirle a la aplicación que deberá representar en cada uno de los widgets marcados.

Mediante el etiquetado de los componentes podremos ponerle un nombre y asignar las diferentes propiedades que tendrá cada elemento, Figura 11. Dichas propiedades serán diferentes de acuerdo al tipo de widget que deseemos utilizar. No serán las mismas las propiedades que tenga un input de las que tenga una imagen.

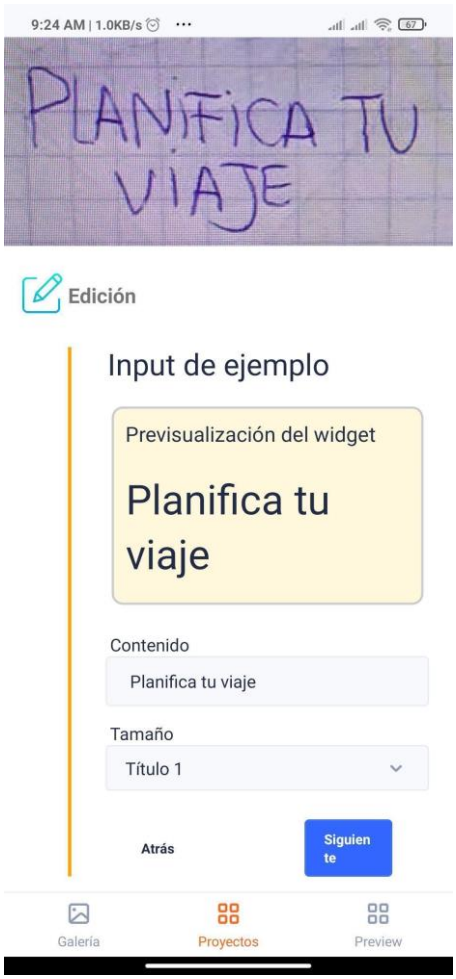

Figura 11 Imagen ilustrativa de la edición de un widget de texto, en este caso las propiedades que se agregan son tamaño y contenido.

4.2.4 Previsualización de los prototipos creados

Una vez digitalizados los diferentes wireframes o pantallas, podremos comprobar cómo será su representación final mediante el módulo de previsualización. En esta etapa podremos verificar si los prototipos se ven como nosotros necesitamos o si es necesario realizar algún cambio.

Es importante destacar que la previsualización nos permite hacer uso de la navegación, en el caso que se estén representando botones u otros widgets los cuales nos permiten redirigir el prototipo hacia otras pantallas. Mediante esta funcionalidad

no solo podremos ver cada pantalla por separado, sino también probar cómo sería el flujo entre las diferentes pantallas de una aplicación.

En la siguiente Figura 12, podemos observar que hay dos botones con navegación: SIGUIENTE y OMITIR. Cada uno de ellos nos podría llevar a diferentes pantallas.

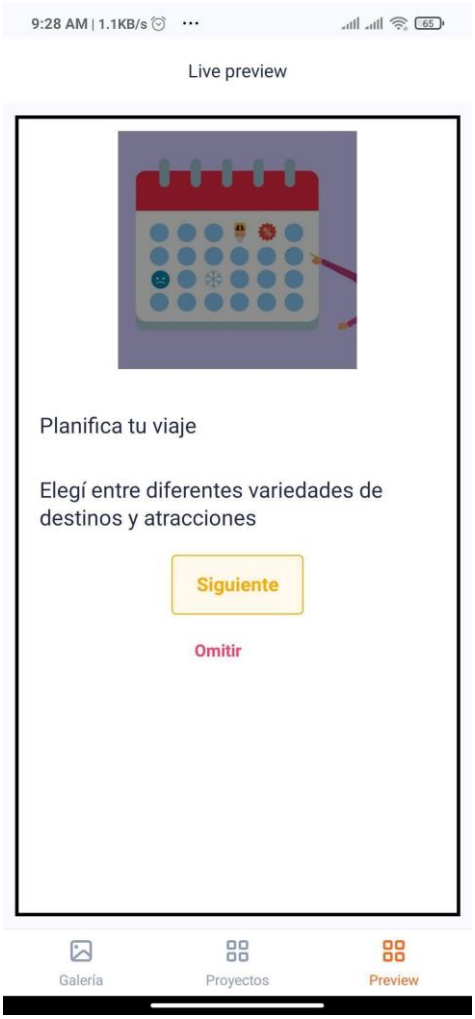

Figura 12 Imagen que muestra la previsualización de un prototipo digitalizado con una imagen, dos textos y dos botones. Los botones tienen navegación hacia otras pantallas

4.2.5 Pruebas de usabilidad

Luego de digitalizados los prototipos en papel, para poder obtener las diferentes métricas de usabilidad se necesitó crear un módulo específico para poder ofrecer dicha funcionalidad. Este módulo es el responsable de gestionar las diferentes pruebas

realizadas por los usuarios finales. Asimismo, es también quien lleva a cabo la medición de los parámetros de usabilidad que nos permiten evaluar los diferentes aspectos de la usabilidad de los prototipos creados.

Entre las funcionalidades principales de esta parte se encuentran la creación de nuevas pruebas de usabilidad, el desarrollo de un mecanismo para poder ejecutar cada prueba junto con la gestión completa del ciclo de vida de la ejecución de cada prueba. Dicho ciclo comienza cuando un usuario arranca una determinada prueba y finaliza cuando el mismo brinda su opinión sobre el prototipo utilizado.

Toda prueba de usabilidad comienza en una determinada pantalla y se espera que finalice en otra diferente. Las pantallas serán definidas por el evaluador al momento de crear la prueba.

Cuando creamos una nueva prueba se establecen diferentes parámetros necesarios para poder crear las mismas. Entre ellos se encuentra el nombre, una pantalla de origen y una pantalla objetivo. En función de estos parámetros se determina en cuál de todas las pantallas comenzará la prueba y en qué pantalla finaliza. Esto es muy útil ya que nos permite brindar mayor flexibilidad al momento de crear las pruebas.

Por otra parte, también existen 2 parámetros adicionales que pueden o no utilizarse. Los dos parámetros adicionales comprenden el tiempo máximo que tendrán los usuarios para completar la prueba y el número mínimo de pantallas requeridas para poder finalizarla. Es decir, si para completar un determinado procedimiento inevitablemente hay que pasar por 4 pantallas distintas, el número mínimo de pantallas requeridas será igual a 4.

Los dos parámetros adicionales, también son definidos por el evaluador y sirven para determinar cuántos usuarios son capaces de completar una tarea en un tiempo determinado, o bien, poder comparar la cantidad total de pantallas visitadas con el número mínimo de pantallas requeridas que los usuarios deberían visitar.

Por último, las pruebas pueden finalizar por 3 motivos diferentes que se listan a continuación:

1. Los usuarios consiguieron llegar a la pantalla objetivo.
2. Se alcanzó el tiempo máximo establecido para finalizar la prueba.

3. Los usuarios eligieron de forma intencionada no concluir la misma. Sin importar el motivo, siempre se puede elegir deliberadamente abortar la prueba pudiendo luego, brindar más detalles sobre los motivos por los cuales fue abortada.

Si las pruebas finalizan por el primer motivo se dice que fueron pruebas exitosas, ya que se logró alcanzar el objetivo deseado. Por el contrario, si finalizó por los otros 2 motivos se considera que la prueba resultó fallida, dado que el usuario no logró completar la tarea planteada en la prueba.

4.3 Descripción de apariencia y diseño

La interfaz de la aplicación MWP ha sido desarrollada para ofrecer una experiencia que se focaliza en un uso sencillo y en un entorno amigable al usuario.

Para ello se centra en tres estrategias de diseño:

1. Interfaz intuitiva.
2. Presentación amigable.
3. Maleabilidad y dinamismo.

A continuación, en la Figura 13, se podrá visualizar los primeros wireframes que planteamos como solución. Comenzada la implementación, se fueron mejorando poco a poco algunos detalles de los wireframes como de las ideas que teníamos en un primer momento para que sean realmente efectivas y así poder cumplir con nuestro fin, aunque podemos decir que en general siguen siendo realmente similares los primeros wireframes a la versión de la aplicación actual.

La idea de la aplicación MWP, además que sea útil para la digitalización, es mantener la facilidad con la que los usuarios pueden usarla y mantener un alto nivel de satisfacción por parte de los usuarios en el momento del uso de la aplicación. Es por eso que las únicas diferencias que se encontrarán en la versión actual de la aplicación son mejoras.

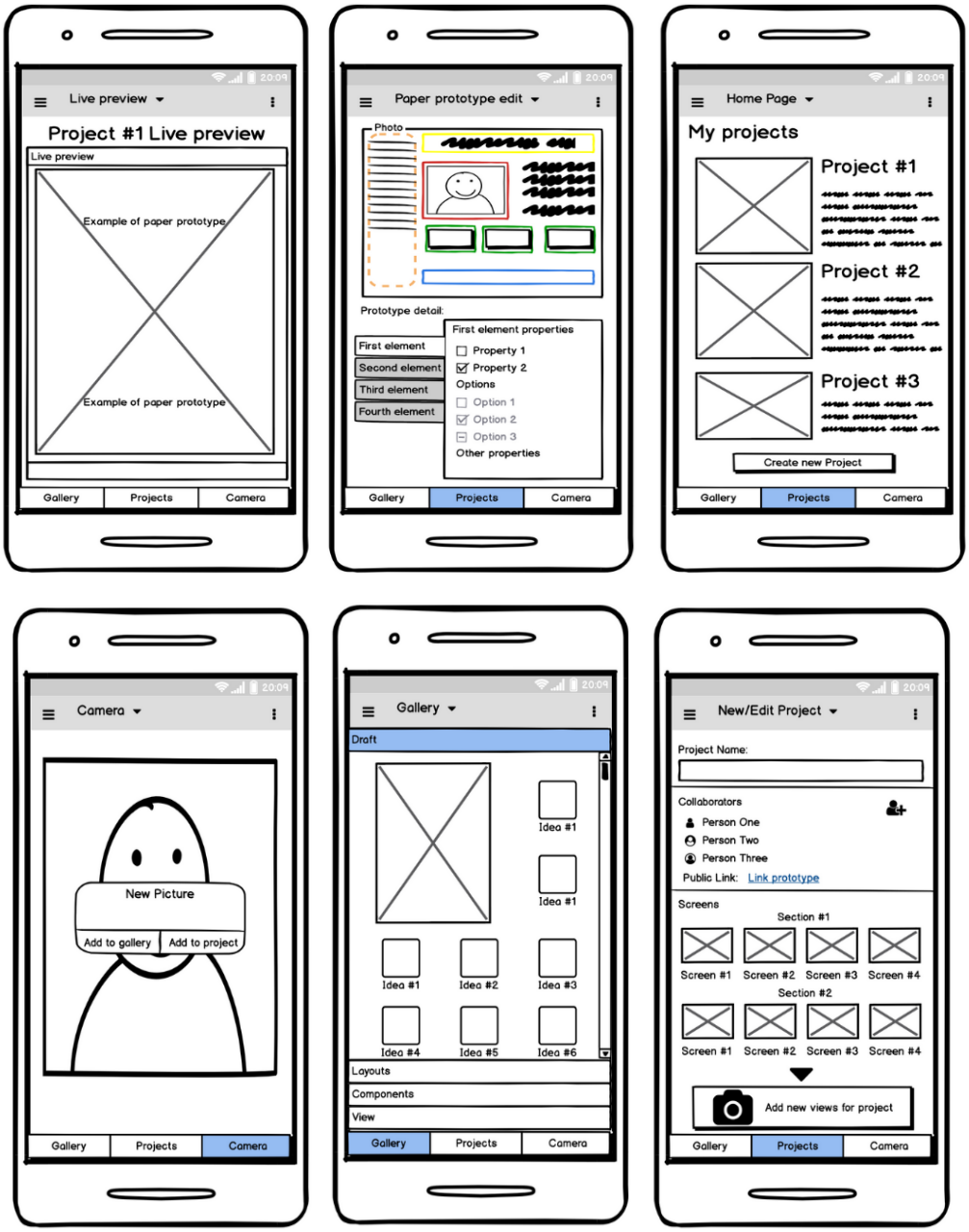

Figura 13 Wireframes iniciales de la aplicación MWP

4.3.1 Interfaz intuitiva

El diseño propone una interfaz donde se despliegan secuencialmente los distintos elementos que la constituyen, Figura 14, ordenados según la lectura occidental de izquierda a derecha, facilitando así realizar los distintos pasos o visitar las páginas deseadas con comodidad y de manera sencilla.

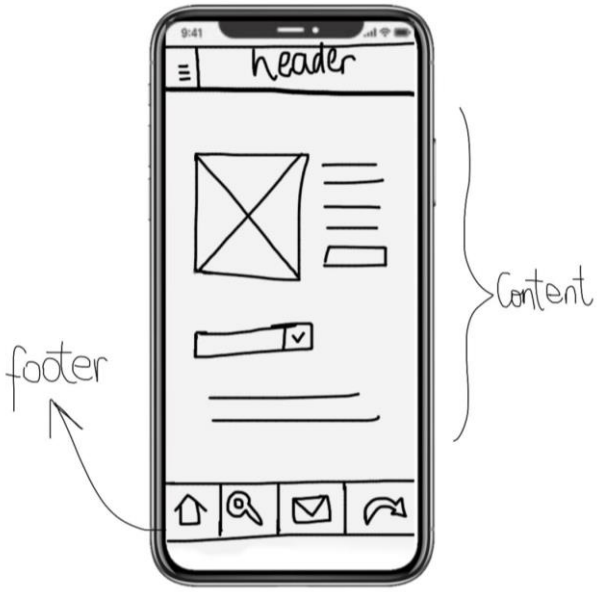

Figura 14 Wireframe de una página con las principales secciones.

El encabezado nos ubicará en qué parte de la aplicación nos encontramos actualmente, la zona del cuerpo de la aplicación reúne la totalidad de las interacciones y por último sobre el margen inferior, Figura 15, de la aplicación nos encontraremos con los accesos directos a diferentes partes de la aplicación de gran importancia.

Figura 15 Barra de navegación inferior.

4.3.2 Presentación amigable

El diseño presenta una visión que permite sentirse en un entorno despojado que invite a concentrarse en la digitalización de los prototipos. Para ello se recurre al uso de una paleta cromática donde predomina el color azul, y se reserva el uso de los opuestos (el rojo) para remarcar la opción de cancelar una operación.

La paleta es fría, y la interfaz tiende a ser minimalista, esto facilita la identificación, por parte del usuario, de un espacio neutro que evita la sobrecarga de signos y apuntala la sensación de facilidad de uso.

Así una interfaz amigable propone una relación directa del usuario con el contenido a través de una única página que evita la redundancia y facilita los índices de navegación.

4.3.3 Maleabilidad y dinamismo

La propuesta recurre a una estrategia asincrónica de manera que el usuario no tenga, ante las distintas operaciones, que recargar la página. Fuera de las conocidas ventajas: simplicidad, velocidad de interacción, permanencia del entorno, este uso tiene como principal inconveniente la sensación estática que suele generar.

A fin de evitar este problema, los cambios que se realicen podrán ser visualizados inmediatamente en la aplicación.

Capítulo 5: Ejecución de la metodología para evaluación de usabilidad en prototipos en papel

En este capítulo, llevaremos adelante la ejecución de la metodología para la evaluación de usabilidad de manera temprana que detallamos en el Capítulo 3: Veremos cómo podemos realizar diferentes pruebas de usabilidad dentro de un proyecto y cómo podemos obtener información relevante para poder hacer mediciones de usabilidad. De esta manera, podremos conseguir datos que resulten de importancia para entender si el diseño de una aplicación es el correcto o si existen problemas de usabilidad sobre los cuales se puedan realizar mejoras.

Para llevar adelante la ejecución de la metodología, utilizaremos el proyecto *"Finde Largo"*: una aplicación diseñada para la visualización y reserva de hospedajes. Mediante este proyecto de ejemplo podemos representar cada una de las 4 etapas que componen la metodología, comenzando con la construcción de los prototipos en papel, hasta la interpretación de los resultados obtenidos en las métricas de usabilidad.

En el primer paso de la metodología realizaremos los wireframes de las pantallas de interés para el proyecto. A continuación, digitalizaremos los wireframes. Utilizando la aplicación diseñada para tal fin, tomaremos una foto e identificaremos los widgets.

Luego de concluida la digitalización, en el tercer paso se decidirán los casos de uso a evaluar, dando prioridad a los más importantes. Habiendo tomado la decisión sobre los casos de uso prioritarios y las métricas de usabilidad a evaluar, realizaremos las pruebas, para luego, en el cuarto y último paso proceder a interpretar los resultados obtenidos al finalizar la ejecución de las pruebas.

5.1 Primer paso: Construcción de los prototipos en papel

Comenzaremos por definir y dibujar cada uno de los wireframes correspondientes a el flujo que tendrá la aplicación. En la siguiente imagen, Figura 16, podemos observar los wireframes correspondientes a la sección *Bienvenida* del proyecto *"Finde Largo"* que estará compuesta por tres pantallas que se mostrarán para todos los usuarios la primera vez que utilicen el prototipo. La secuencia de imágenes se podrá visualizar una detrás de la otra si se hace clic sobre el botón "Siguiente", de todas formas, si un usuario ya se encuentra familiarizado con la aplicación, la secuencia de imágenes podrá ser ignorada.

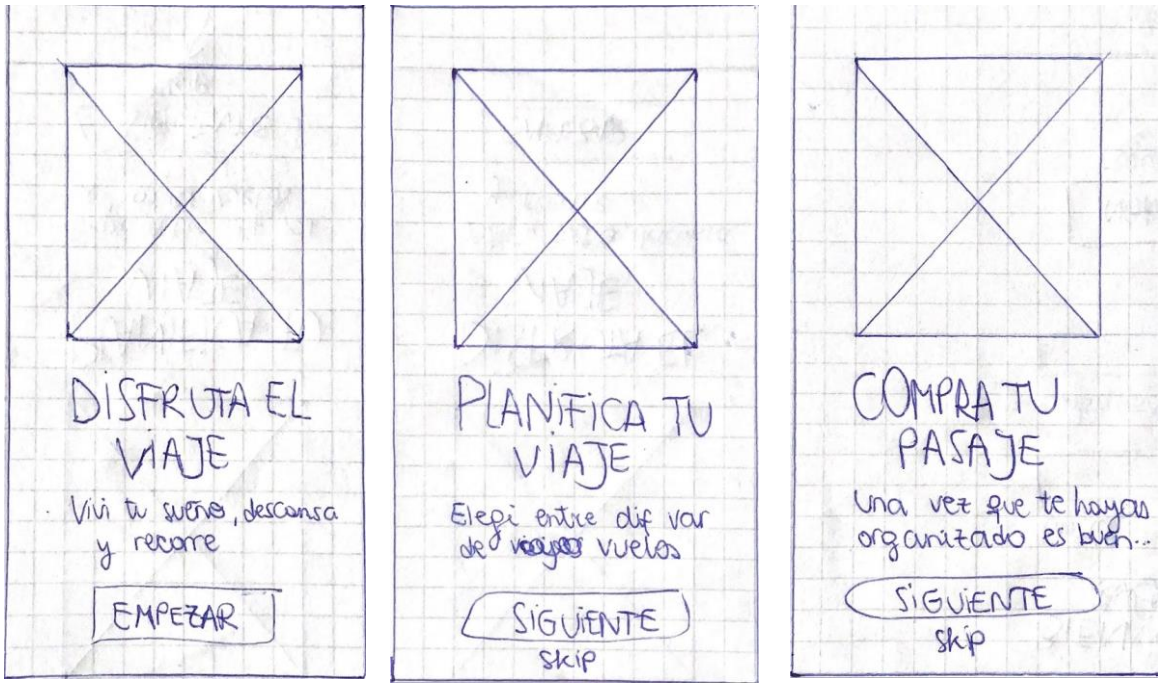

Figura 16 Flujo de bienvenida.

Una vez finalizadas las pantallas de bienvenida, nos encontraremos en la página principal donde se muestra un listado con la información de nuestra aplicación, Figura 17. La lista contiene el listado de sitios, junto a una imagen representativa, un título y una descripción. Haciendo clic sobre uno de los ítems listados, nos dirigiremos al detalle del mismo.

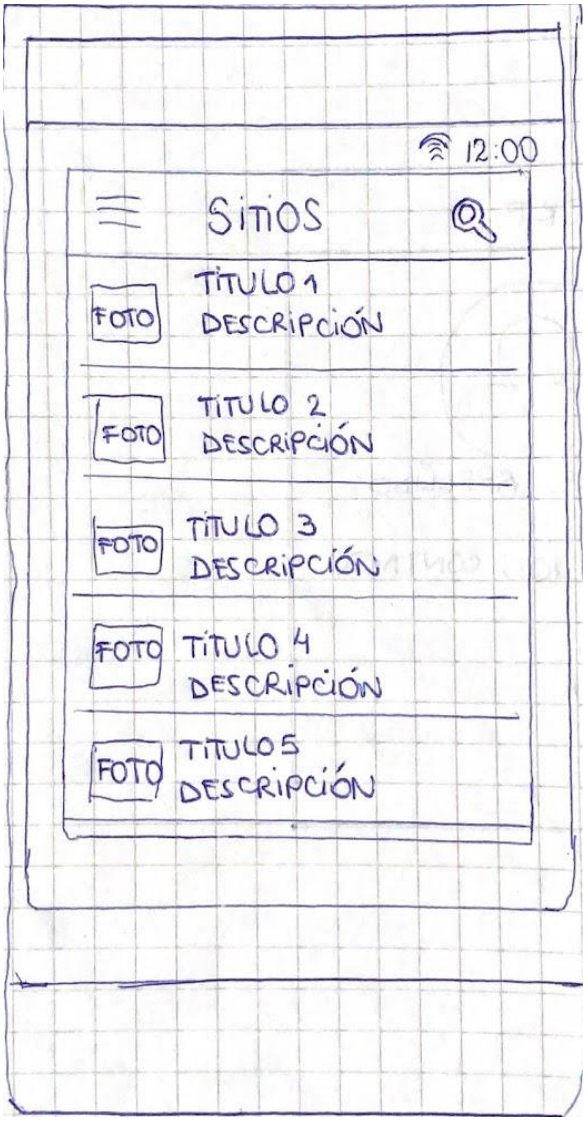

Figura 17 Página principal de la aplicación.

Dentro de la vista del detalle del ítem, Figura 18, podremos ver también una imagen, su nombre, descripción, como así también un listado de fechas disponibles y un botón para confirmar la reserva de un hospedaje.

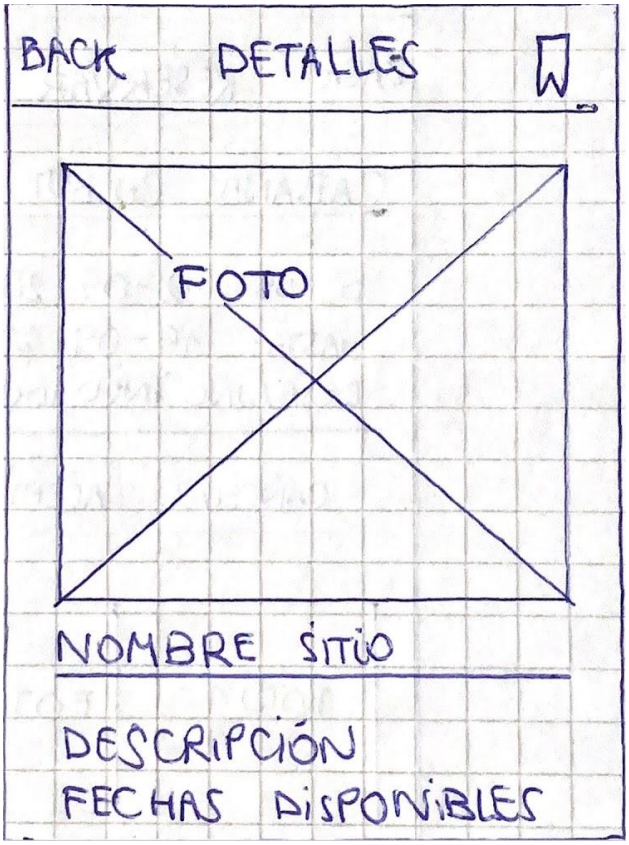

Figura 18 Vista del detalle.

5.2 Segundo paso: Digitalización de los prototipos

Planteamos una serie de casos de uso para el correcto funcionamiento de la aplicación en los que se puede observar el paso a paso de la digitalización de los diferentes wireframes.

Comenzando con la creación de un proyecto, capturando la imagen y asignándole a una determinada sección del proyecto hasta llegar a la vista de la pre visualización con la navegación a través de los widgets digitalizados.

5.2.1 Crear proyecto

En este primer paso vamos a crear un nuevo proyecto desde la página principal de la aplicación. Para ello, se ingresa el **nombre del proyecto** y una breve **descripción**.

5.2.2 Agregar un wireframe a un proyecto

El segundo paso consiste en agregar un nuevo wireframe al proyecto. Un wireframe podrá ser agregado desde la vista de detalle del proyecto haciendo click sobre el botón azul *"Agregar link"*, para luego asignarle un nombre y la imagen que corresponderá a ese wireframe, que vamos a digitalizar. También se desplegará un select para que seleccionemos la sección del proyecto a la cual pertenecerá.

La imagen a agregar podrá ser tomada en el momento seleccionando la opción que corresponda como se muestra en la Figura 19. También tendremos la opción de utilizar una imagen ya existente o agregarla desde la galería. Una vez seleccionada y editada la imagen, la podremos visualizar en el modal *"Agregar nuevo wireframe"*.

Figura 19 Modal con la información a completar para el nuevo wireframe.

5.2.3 Edición de la grilla de componentes

Explicaremos cómo se debe realizar la edición de marcado y grilla desde una imagen agregada con anticipación. Para ello se podrá realizar una grilla sobre la cual se ubicarán los componentes visuales que contendrá el prototipo.

Partiremos desde la imagen del prototipo en papel, Figura 20, y sobre ella podremos agregar diferentes guías que nos permiten configurar el tamaño de cada uno de los componentes que tendrá cada pantalla.

Las guías pueden ser verticales (azules) y horizontales (rojas). Nos permitirán identificar el componente que se digitalizará para la previsualización, Figura 21. Estas guías pueden ser agregadas o eliminadas desde los botones del margen inferior.

Figura 20 Líneas de trazado verticales y horizontales.

Para poder eliminar una guía tendremos que seleccionar la guía que deseamos eliminar previamente, haciendo un click sobre ésta. Las guías pueden moverse a lo largo y ancho de la imagen.

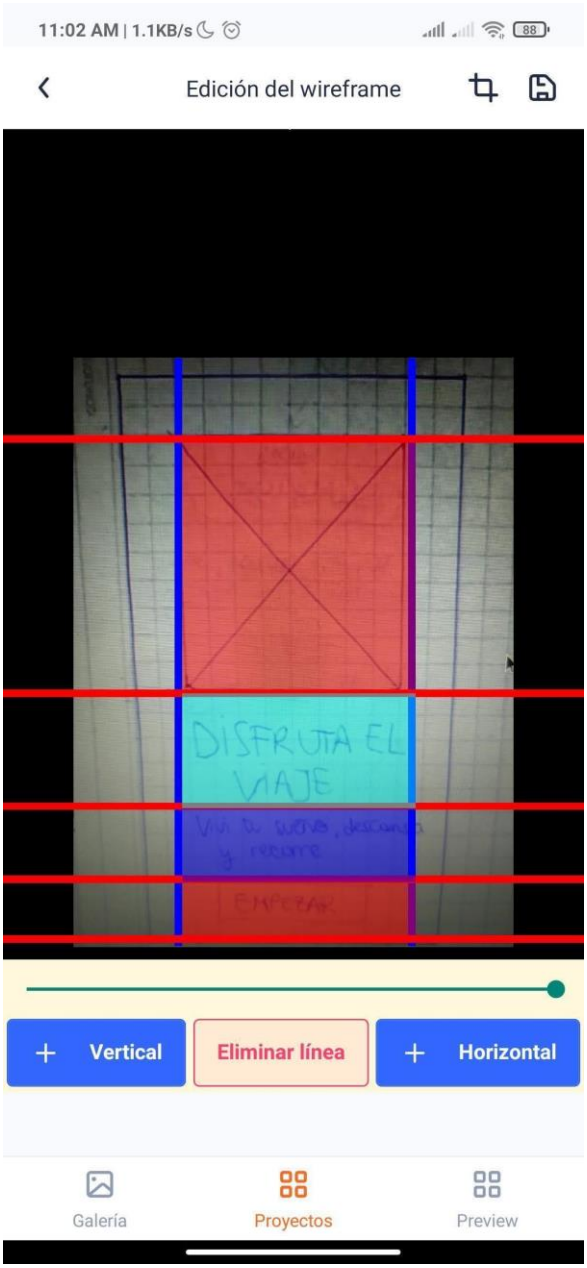

Figura 21 Trazado de las líneas para digitalizar el componente seleccionado.

En la imagen anterior podemos observar al conjunto de guías junto con algunas áreas de la imagen en diferentes colores. Cada una de esas secciones con color sobre la

imagen del prototipo en papel representarán determinados widgets que se deberán representar en el prototipo digitalizado.

Una vez finalizado el marcado del componente, Figura 21, procedemos a seleccionar sobre el icono **Guardar** que se aloja en el margen superior derecho.

5.2.4 Determinar los atributos en los wireframes

En este caso explicaremos el paso a paso a seguir para poder asignar propiedades a un widget en particular. Esto lo realizaremos desde la pantalla de *Edición de Widget*.

En el primer paso del wizard **"Elegir componente"** que aparece en la figura 22, seleccionaremos desde el menú de componentes el widget que vamos a representar. Además, le agregaremos un nombre descriptivo que será utilizado en el futuro y haremos click en "Siguiente".

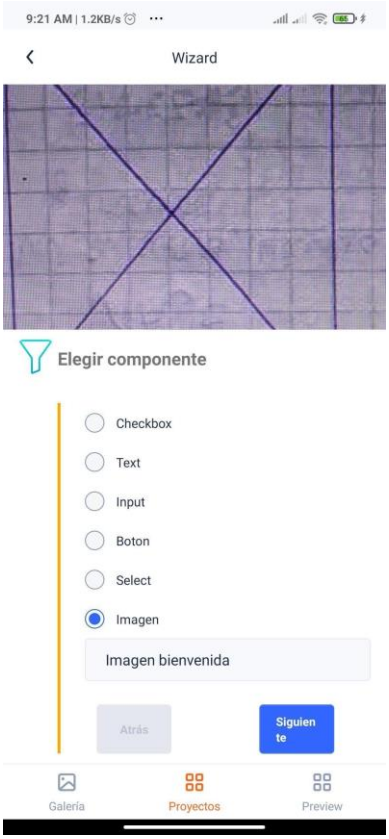

Figura 22 Primer paso, lista de componentes para utilizar en la digitalización.

Aclaración: El botón siguiente del paso “Elegir componente” **permanecerá deshabilitado** hasta que no se haya seleccionado el widget desde el menú y se haya escrito su nombre siendo ambas acciones requeridas para avanzar hacia el siguiente paso del wizard.

Una vez en el segundo paso, Figura 23, comenzaremos seleccionando las propiedades que tendrá nuestro componente. Para ejemplificar con una imagen, a continuación, procederemos a seleccionar el componente BOTÓN.

Figura 23 Segundo paso, selección de propiedades para el botón “Siguiente”.

Dentro de las propiedades que se pueden modificar se encuentra el color, el tamaño, el tipo de botón que utilizaremos y el texto que contendrá. Los cambios se

verán reflejados en el sector amarillo de la previsualización del widget, representación tal cual va a quedar el botón una vez que finalicemos el proceso de edición.

En este paso del wizard, vale aclarar que **las propiedades que se pueden modificar son diferentes para cada uno de los componentes** que se listan en el menú del primer paso.

Para confirmar la edición del componente, Figura 24, seleccionaremos a qué wireframe será agregado para que luego sea utilizado en la previsualización del mismo.

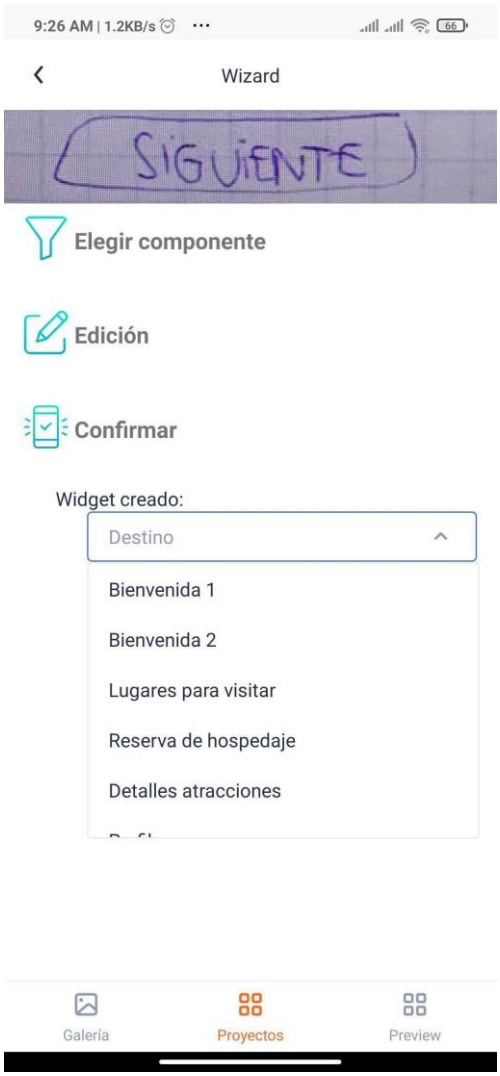

Figura 24 Tercer paso, selección del widget donde se agrega el componente digitalizado.

Por último, haremos click en el botón *"Aceptar"* o cancelaremos la edición desde el botón *"Cancelar"*.

5.2.5 Previsualización

En este paso se puede observar cómo es la previsualización de un wireframe en papel, Figura 25. Para la digitalización del wireframe, el usuario tuvo que haber editado cada uno de los componentes que conforman la siguiente imagen.

Entre los componentes más utilizados se pueden encontrar: imágenes, subtítulos y títulos.

Figura 25 Pre visualización de una vista.

5.3 Tercer paso: Definir y realizar las pruebas de usabilidad con usuarios

5.3.1 Definir las pruebas de usabilidad

Cuando pensamos en una plataforma para reserva de viajes existen muchas funcionalidades que podrían resultar importantes. Antes de continuar avanzando, es importante definir el alcance del proyecto y determinar cuáles historias de usuario serán contempladas en las pruebas de usabilidad y cuáles no. Continuando con lo descrito en la sección anterior, en este ejemplo puntual se ha decidido probar la bienvenida y la reserva de hospedajes. A partir de este caso de uso concreto, sobre el cual se desea medir la usabilidad, es muy importante considerar cuáles son los atributos de usabilidad que deseamos comprobar y validar a partir de las pruebas con los usuarios finales.

5.3.2 Creación y desarrollo de las pruebas de usabilidad

Luego de concluido el segundo paso, y finalizada la digitalización de los prototipos podremos determinar que pruebas de usabilidad necesitamos incluir en nuestro proyecto, como se muestra en la Figura 26. Para ello, deberemos crear tantas pruebas de usabilidad como casos de uso quisiéramos evaluar.

Para poder crear las pruebas de usabilidad, dentro de un proyecto determinado presionaremos en el botón *“Pruebas de usabilidad”*, el cual nos llevará a una nueva pantalla donde podremos crear pruebas nuevas.

Figura 26 Creación de una prueba de usabilidad

A cada una de las pruebas le podremos asignar un nombre, una descripción, la cantidad de pantallas que se deben atravesar para completar la prueba en caso que sea necesario, el tiempo máximo que se dispondrá para completar la prueba o si es indefinido, la pantalla de origen en la que comenzará la prueba y la pantalla destino la cual ocasionará la finalización de la prueba.

Una vez completados los datos de la pantalla podemos optar por la confirmación de la prueba o cancelarla.

Figura 27 Captura de pantalla de las vistas de pruebas de usabilidad.

La prueba puede finalizar por diversos motivos, entre los cuales encontramos: la expiración del tiempo definido en la pantalla anterior, la llegada del usuario a la pantalla destino con éxito o porque se decidió abortar la prueba.

Independientemente del motivo de finalización, una vez que la prueba haya concluido, Figura 27, veremos en pantalla la opción de calificación que deberemos completar con nombre, responder a las preguntas y asignar una calificación para cada una de las mismas y también será posible escribir un comentario que permitan conocer en profundidad la percepción sobre el prototipo evaluado.

Daremos por finalizado el proceso de evaluación de una prueba cuando se haya calificado y enviado la evaluación.

5.4 Cuarto paso: Interpretación de las métricas de usabilidad obtenidas de las pruebas.

En este paso, luego de haber definido y realizado las pruebas de usabilidad, deberemos proceder a la interpretación de los resultados obtenidos. A continuación, se detallan los atributos que pueden ser evaluados, junto con la construcción de un tablero de control para representar los valores de las métricas obtenidas. Este paso es de vital importancia para conocer los resultados que arrojaron las pruebas de usabilidad realizadas con anterioridad.

5.4.1 Información obtenida de las pruebas de usabilidad

Una vez definidos los casos de uso que se quieren analizar, es importante entender cómo se realiza el seguimiento del comportamiento de los usuarios cuando participan de las pruebas de usabilidad. Para ello, dentro de la aplicación móvil, se definieron 4 tipos de eventos diferentes que nos ayudan a medir la usabilidad de un prototipo. Dichos eventos son[8]:

1. Inicio de una prueba de usabilidad: este evento se registra al comenzar con cada una de las pruebas.
2. Navegación de pantallas dentro de la prueba de usabilidad: cada vez que el usuario presiona en alguno de los botones del prototipo y es redirigido hacia otra pantalla se registra un evento de navegación.
3. Finalización de la prueba: una vez que la prueba sea finalizada, se registra el fin de la prueba junto con las diferentes causas por las cuales pudo haber finalizado. Dichas causas podrían ser:
    a. El usuario terminó de forma exitosa la prueba, es decir, llegó a la pantalla de destino.
    b. El usuario decidió abortar la prueba por el motivo que haya considerado oportuno.

[8] Para mayor información sobre la información generada en cada evento refiérase a 6.4.5 , donde se explica en mayor profundidad qué parámetros se envían y cómo se obtienen los mismos

c. La prueba finalizó por haberse cumplido el tiempo máximo destinado para la realización de la misma. Ese tiempo es definido al momento de crear la prueba de usabilidad.

4. Evaluación de la prueba por parte del usuario: en este evento se registran los comentarios, sugerencias o mejoras que el evaluador considere necesario aportar al proyecto junto con las métricas del nivel de satisfacción y facilidad de uso.

5.4.2 Atributos que se pueden evaluar cuando realizamos pruebas de usabilidad

5.4.2.1 Medición de Efectividad

Si quisiéramos medir la efectividad de nuestros prototipos mediante el uso de la aplicación desarrollada, podríamos comprobar el porcentaje de tareas completadas con éxito al primer intento (E) o las Tareas resueltas en un tiempo limitado (TTL). Para eso deberemos considerar el estado en el cual los usuarios finalizan la prueba, pudiendo este estado contener 3 valores: Abortado, Finalizado exitosamente o finalizado por exceder el tiempo límite para realizar la prueba.

Considerando los posibles estados por los cuales puede finalizar la prueba, para poder medir el porcentaje de tareas completadas con éxito al primer intento deberemos aplicar la siguiente fórmula:

$$E = ps / pr$$

E =Porcentaje de tareas completadas con éxito en el primer intento (1)

ps = Cantidad total de pruebas que finalizaron satisfactoriamente

pr = cantidad total de pruebas realizadas

Por otra parte, para medir las tareas resueltas en un tiempo limitado (TTL), deberemos utilizar la siguiente fórmula

$$TTL = ps / pna$$

E = Porcentaje de tareas completadas con éxito en el primer intento

pa = Cantidad total de pruebas abortadas

$$pna = ps - pa$$

5.4.2.2 Medición de Eficiencia

La eficiencia se puede medir a través de diferentes atributos. Mediante la utilización de la aplicación podremos conocer algunos como el *tiempo* empleado para completar una tarea (TT), el tiempo transcurrido en cada pantalla (TP) o la eficacia relativa en comparación con un usuario experto (EUE).

Si queremos calcular el *tiempo* empleado para completar una tarea (TT) deberemos analizar los eventos de finalización de las pruebas. En dichos eventos existe la variable *total_testing_time* la cual representa el tiempo total en milisegundos que el usuario demora para finalizar la prueba.

A su vez, deberemos filtrar dichos eventos sólo por aquellos eventos en los cuales la prueba realizada haya finalizado de manera exitosa. Es decir, que la variable *testing_status* contenga el valor de finalización de la prueba de manera correcta. La fórmula para calcular esta métrica es:

$$TT(conjunto\_eventos) = i = 1ps(ttt(evento\_finalizado(i)) / ps$$

ttt = función que retorna el valor de *total_testing_time* de un evento dado

Por otro lado, si quisiéramos calcular la eficacia relativa en comparación con un usuario experto (EUE) deberemos comparar el tiempo que un usuario experimentado le toma realizar una tarea (TT) en comparación con el tiempo que a un usuario nuevo

$$EUE = TT(usuarios\_expertos) / TT(usuarios\_nuevos)$$

usuarios_expertos = eventos de tipo testing_ends con parámetro first_time false

usuarios_nuevos = eventos de tipo testing_ends con parámetro first_time true

Al momento de evaluar la eficacia relativa en comparación con un usuario experto (EUE) es importante entender los posibles valores que podemos obtener. Si obtenemos un valor menor a 1, podremos verificar que los usuarios experimentados tienen un mayor nivel de eficiencia respecto a los usuarios que utilizan los prototipos por primera vez.

En cambio, si el valor obtenido es mayor a 1 nos indicará que los mismos tardan más en utilizar el prototipo y por lo tanto se reduce su eficiencia. Si se da este caso, será

importante poder realizar cambios para que los usuarios experimentados puedan mantener la eficiencia a lo largo del tiempo.

5.4.2.3 Medición de Satisfacción

La satisfacción puede medirse de acuerdo al nivel de dificultad que ellos perciben al realizar la prueba o bien si los prototipos les agradan o no a los posibles usuarios finales.

Para ello, luego de finalizadas las pruebas se les pide a los participantes que realicen una evaluación de la experiencia que tuvieron durante la prueba mediante 2 preguntas se puede conocer qué tan fácil les resultó la prueba que tuvieron que realizar y si el prototipo presentado es agradable.

Dichas preguntas tienen una escala de 1 a 5 en donde se busca conseguir el mayor puntaje posible.

5.4.2.4 Medición de Memorabilidad

Cuando un usuario comienza a interactuar con un sistema, es muy factible que durante las primeras veces que lo utilice pueda cometer errores al momento de interactuar con el mismo para realizar una tarea. En una misma pantalla pueden existir múltiples links, íconos o botones que cambian el contenido que los usuarios visualizan en la interfaz gráfica.

Para poder medir la memorabilidad podemos comparar y analizar la cantidad total de pantallas que los usuarios utilizan para poder completar una tarea como así también cuales de las pantallas visitan. Al momento que creamos una nueva prueba de usabilidad, podemos establecer una cantidad mínima de pantallas que es necesario atravesar para poder completar la prueba. Sin embargo, una aplicación tiene muchas pantallas diferentes donde cada una puede llevar a distintos lugares. En el caso de querer evaluar la memorabilidad, podríamos analizar la cantidad de pantallas visitadas que los usuarios que ya utilizaron el prototipo visitan, en comparación con ese mismo valor para los usuarios nuevos.

Un usuario experimentado podría, en este ejemplo completo, omitir las pantallas de bienvenida porque ya conoce la aplicación. En cambio, un usuario que

nunca utilizó el prototipo es más probable que necesite ejecutar el proceso de *Bienvenida* completo e incluso pueda cometer errores durante la interacción con el prototipo.

De esta manera, analizando los eventos *testing_screen_navigation* podremos conocer por cuáles de las pantallas disponibles en los prototipos los usuarios navegan. Lo esperable en esta situación es que los usuarios experimentados solamente visiten las pantallas correctas que están directamente relacionadas con la prueba en la cual están participando, mientras que los usuarios nuevos podrían visitar mayor cantidad de pantallas al visitar todas las pantallas de *Bienvenida*.

A menor cantidad de errores de navegación los usuarios experimentados sufran, podremos decir que la memorabilidad es mayor. Por el contrario, si los usuarios ya experimentados con los prototipos cometen errores o se dirigen a pantallas que no se relacionan con la prueba podremos determinar que el nivel de memorabilidad es menor.

5.4.2.5 Medición de distintos atributos adaptados al contexto de negocio

Si bien es posible analizar la usabilidad en múltiples factores como mencionamos antes, también es importante pensar cómo la usabilidad se puede relacionar con determinados objetivos de negocio, que dependen del contexto para el cual se esté trabajando. Cada organización tiene sus propios objetivos y es importante que nuestros prototipos puedan responder y adaptarse a las necesidades particulares de cada proyecto.

Es en este momento donde hay que pensar en la relación entre el prototipo que se generó en el primer paso junto con los objetivos para los cuales fue concebido. Para eso vamos a continuar trabajando en el caso de uso presentado anteriormente.

Si pensamos en el flujo de bienvenida y reserva de hospedajes hay varias preguntas que resultan importantes pensar antes de avanzar con las pruebas. Algunas de estas preguntas podrían ser:

> ¿Cuáles son los atributos específicos que definen la efectividad en el contexto de este prototipo?

¿Qué importancia le dan los usuarios a las pantallas de bienvenida que ven cuando comienzan a utilizar el prototipo? ¿Desean visitar las mismas o prefieren omitir el mismo porque no consideran que les aporte valor?

¿Cómo afectan esas primeras pantallas a las posibilidades de conseguir que los usuarios realicen de manera efectiva una reserva?

En función de estas u otras posibles preguntas se deben desarrollar las pruebas de usabilidad. Es importante pensar qué información nos gustaría obtener de las pruebas y en función de eso se podrá determinar cómo podríamos conseguir esa información. Este apartado deja de ser general para cualquier producto y se convierte en un análisis particular aplicado a los diferentes contextos de negocio.

En respuesta a los interrogantes anteriores podríamos decir que la efectividad para este caso de uso podría medirse como el porcentaje de usuarios que pudieron completar la prueba, es decir, hacer efectivamente la reserva, independientemente si luego de pasar por el proceso de inducción. En relación a dicho proceso podríamos evaluar cuántos de ellos finalizaron todos los pasos de bienvenida y cuantos prefirieron omitir dichas pantallas. En este caso de uso, al existir un flujo secuencial de las pantallas, fácilmente podremos determinar si el usuario pasó o no por todas.

Por otra parte, si quisiéramos evaluar que tan efectivo es nuestro flujo de inducción para conseguir que los usuarios realicen la reserva podríamos comparar el porcentaje de usuarios que finalizó exitosamente la prueba de usabilidad, considerando solo aquellos que participaron de la inducción completa, distinguiéndolos de aquellos que no. De esta manera, podríamos conocer cómo afecta la bienvenida a la efectividad de nuestro proyecto.

A su vez, como otra prueba diferente, también podríamos arrancar la reserva de un hospedaje directamente dentro del listado de sitios y observar cómo diferentes puntos de inicio podrían afectar a los resultados de nuestra prueba. De esta forma vemos cómo se pueden evaluar diferentes escenarios ya sea en una única prueba o bien mediante pruebas diferentes las cuales podremos combinar para conseguir un resultado conjunto entre sí al combinar las pruebas.

5.4.3 Construcción de un tablero de control para analizar los resultados obtenidos de las métricas

Dado que el objetivo de esta tesina no es hacer pruebas de usabilidad, sino ofrecer una metodología que permita realizar las mismas de manera temprana, en esta etapa no se realizaron las pruebas con posibles usuarios finales. Sin embargo, para poder probar la aplicación creada se realizaron diferentes pruebas de integración, que nos permitieran comprobar si las pruebas de usabilidad y el correspondiente seguimiento de eventos y métricas se estaba haciendo correctamente. Para ello, de manera local durante el proceso de desarrollo, se crearon y ejecutaron diferentes pruebas de usabilidad internas que nos permitieran validar el funcionamiento de la aplicación, pero sin contar con usuarios externos.

La idea en este punto es mostrar cómo una vez concluidas las pruebas podremos visualizar, representar, analizar e interpretar los valores obtenidos de las pruebas.

Para ayudar con la interpretación de las métricas recolectadas durante las pruebas de usabilidad, se desarrolló un tablero de control, el cual nos permitió visualizar y agrupar cada uno de los distintos valores de las métricas de usabilidad obtenidos en las distintas ejecuciones de las pruebas.

En dicho tablero, Figura 28, pudimos representar las diferentes métricas de usabilidad que nos encontramos, junto con diferentes valores, tablas y gráficos que nos permitieron comprender mejor cómo los usuarios se comportan al momento de participar de las pruebas. La información de dicho tablero se obtendrá luego de la ejecución de diferentes pruebas de usabilidad dentro de nuestra aplicación.

A medida que los diferentes usuarios van participando de las pruebas, los resultados obtenidos de las mismas son contabilizados por Google Analytics[9] aportando mayor cantidad de información y nuevos datos al tablero. En la sección 6.4.5 se explica con mayor nivel de detalles cómo fueron implementadas las métricas de usabilidad.

[9] https://analytics.google.com

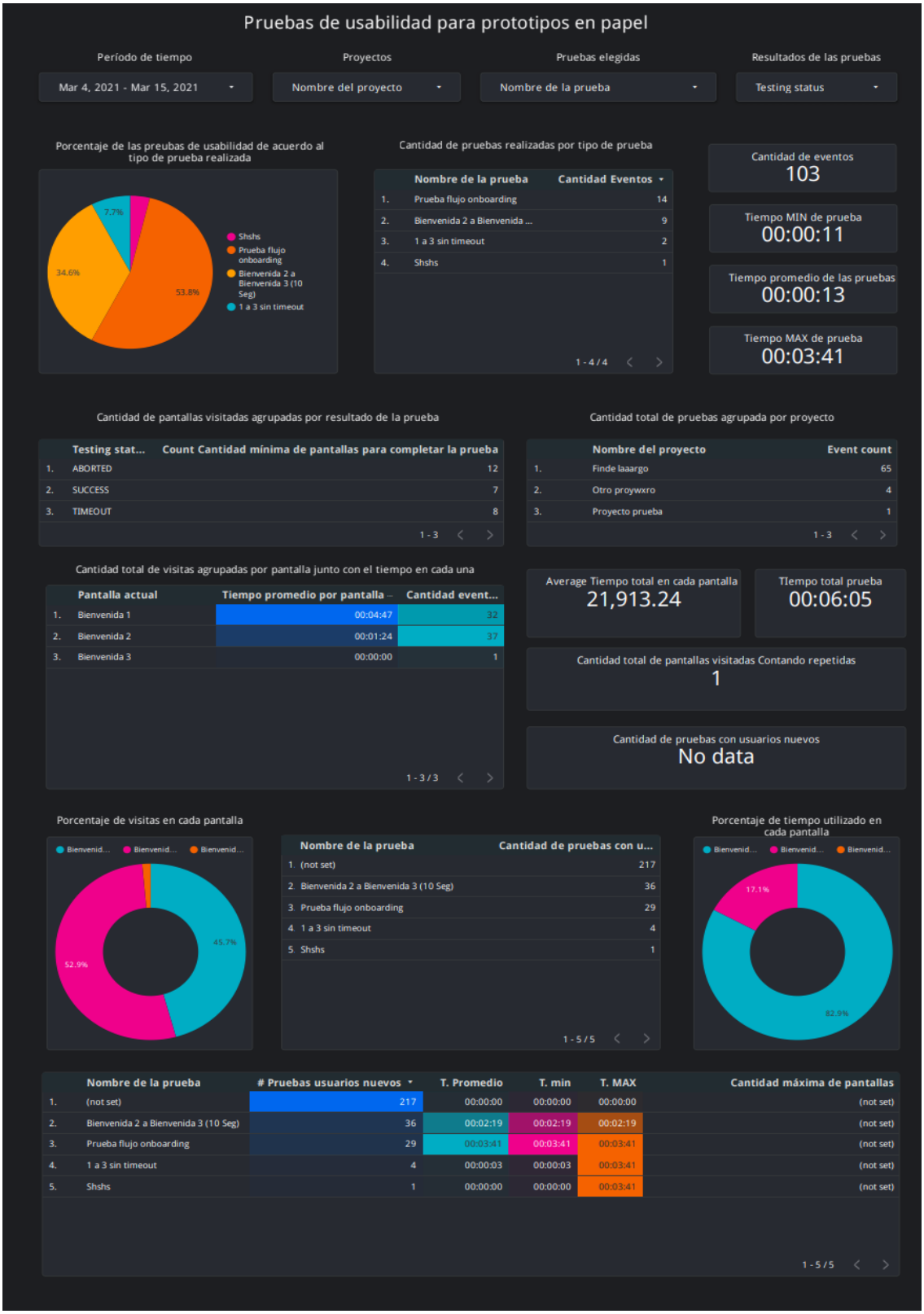

Figura 28 Tablero para representar las diferentes métricas de usabilidad

Anteriormente, en la sección 2.2 hablamos de la importancia de poder obtener métricas de usabilidad y cómo estas podrían ser útiles para ayudarnos a realizar un mejor análisis y tomar decisiones más acertadas con respecto a la usabilidad de una aplicación. También mencionamos muchas métricas sumamente útiles que pueden brindarnos valiosa información acerca de ciertos parámetros de usabilidad de una aplicación.

A través del desarrollo de distintas pruebas de usabilidad de los prototipos en papel digitalizados en nuestra aplicación, podemos obtener información como el porcentaje de tareas completadas con éxito al primer intento, el tiempo empleado en completar una tarea, Tiempo transcurrido en cada pantalla, Nivel de dificultad o el tiempo usado para terminar una tarea la primera vez.

Si observamos el tablero, podemos encontrar muchos datos diferentes entre los que se encuentran algunos filtros, gráficos y tablas, así como diferentes datos que podrían ser de utilidad como el tiempo mínimo, máximo y promedio de las pruebas realizadas o la cantidad total de pantallas visitadas por los usuarios.

Mediante la utilización de los 4 filtros que se encuentran en la parte superior, podremos restringir cuales datos queremos observar para así focalizar la atención en ciertos resultados específicos. Dichos filtros nos permiten limitar los resultados presentados a un proyecto o período de tiempo determinado, como así también a una prueba específica o a las diferentes causas de finalización de la evaluación.

Todos los datos que se observan en la parte inferior del reporte dependen completamente del resultado de los filtros aplicados previamente, es decir, son dependientes de ellos. Si cambiamos los valores de los filtros los datos representados por debajo de ellos se verán afectados también.

En el panel podremos obtener información sobre diferentes dimensiones de análisis, entre las que se encuentran:

- Distribución y cantidad de usuarios que participaron en cada una de las diferentes pruebas.
- Tiempo mínimo, máximo y promedio que necesitaron los usuarios para finalizar las pruebas.
- Cantidad de pruebas que se hicieron en cada proyecto.
- El tiempo promedio que se empleó en cada pantalla.

- Diferentes valores asociados las pruebas realizadas tales como la cantidad de usuarios que ejecutaron la prueba por primera vez, como así también el tiempo promedio para completar la prueba o bien la cantidad media de pantallas visitadas que los usuarios usan para poder realizar la prueba.

5.4.4 Conclusiones sobre el análisis de las métricas obtenidas

Como pudimos observar, mediante el desarrollo del tablero podemos visualizar de manera concreta y sistematizada los resultados en cada una de las pruebas de usabilidad ejecutadas en función de los objetivos establecidos en el segundo paso.

A su vez, este tablero busca presentar métricas que no dependan de un caso de uso en particular. Es por ello que en esta oportunidad nosotros solamente definimos algunas dimensiones y métricas generales que consideramos útiles representar y fueran significativas para una amplia gama de proyectos.

Sin embargo, con toda la información recolectada por las pruebas[10] podrían realizarse otras representaciones diferentes, acorde a las necesidades de los diferentes proyectos o pruebas de usabilidad. Si fuera necesario, el mismo tablero puede continuar ampliándose para cubrir nuevas métricas y casos de uso como los definidas en la sección 5.4.2.5 En dicha sección se desarrollaron métricas personalizadas para el flujo de bienvenida y reserva de hospedajes.

Para este ejemplo no fueron desarrolladas ya que no consideramos importante para la compresión de la metodología poder focalizar en ejemplos tan específicos ligados a un determinado contexto de negocio, pero de ser necesario podrían crearse con facilidad nuevos gráficos que permitan analizar métricas personalizadas.

[10] Para más información respecto a los datos recolectados por las pruebas referir a la sección 5.4.1 Parámetros enviados en los diferentes eventos para poder contabilizar las pruebas

Capítulo 6: Desarrollo e implementación

A lo largo de este capítulo, veremos cómo ha sido el proceso para poder llevar adelante la implementación de la aplicación para digitalizar prototipos en papel.

Entenderemos mejor sobre las tecnologías utilizadas, los motivos por los cuales elegimos las mismas, que otros lenguajes y librerías tuvimos en cuenta como posibles alternativas para realizar el desarrollo del aplicativo, como así también, una descripción tanto de la arquitectura global del desarrollo y de las diferentes partes del software que consideramos más importantes.

En la primera sección, mencionaremos las diferentes alternativas y opciones disponibles para construir la aplicación. Luego pasaremos a explicar las tecnologías seleccionadas para desarrollar la aplicación de nuestra tesina. Por último, explicaremos en detalle la arquitectura de la aplicación, la forma en la que elegimos organizar el código, las diferentes librerías utilizadas y daremos algunos ejemplos de las partes más importantes como puede llegar a ser el sistema de grillas utilizado para el etiquetado de los widgets y para la representación de los wireframes digitalizados.

6.1 Elección del lenguaje y framework utilizado

Cuando hablamos del desarrollo de aplicaciones móviles existe una amplia variedad de opciones para realizar el desarrollo de las mismas. Existen opciones nativas para cada plataforma como así diferentes frameworks multiplataforma que nos permiten desarrollar aplicaciones que funcionen en más de un sistema operativo al mismo tiempo sin la necesidad de realizar modificaciones en el código.

Previo a enfrentarnos con el desarrollo de nuestra aplicación, nos tomamos un tiempo para tener un panorama claro, debatir entre nosotros. De esta forma, una vez realizado el análisis de las tecnologías disponibles, optamos por una de ellas.

Si bien se planteó la posibilidad de utilizar algún lenguaje de programación nativo como podría ser Kotlin[11] para Android o Swift[12] para el entorno iOS, nos inclinamos por una solución híbrida que nos permitiese realizar el desarrollo multiplataforma.

Entre los frameworks y librerías preseleccionadas se encontraban:

1. *Ionic*
2. *VueJS*
3. *React Native*
4. *Flutter*

Para hacer la investigación nos concentramos en diferentes puntos tales como las necesidades propias de nuestro proyecto, la disponibilidad de documentación, los aportes realizados por la comunidad de desarrolladores en cuanto a librerías que nos ayuden con el desarrollo, también evaluamos algunos ejemplos de aplicaciones desarrolladas con cada uno de estos entre otras variables.

En primera instancia se intentó utilizar el **framework Ionic**, pero luego de investigar bastante sobre las librerías que necesitábamos para implementar, por ejemplo, la captura de gestos, tratamiento de las imágenes, sumado a la falta de

[11] https://kotlinlang.org/
[12] https://developer.apple.com/swift

documentación notamos que había muchas funcionalidades que no estaban desarrolladas y que nos serían indispensables y sumamente útiles para poder avanzar con el desarrollo de la app, motivos por los cuales se descartó este framework.

Seguimos en la búsqueda de una librería/framework que si cumpliera todas las expectativas dentro de la lista que mencionamos previamente y realizamos el mismo análisis para cada uno de los ítems de las posibles opciones.

Entre las diferentes posibilidades, también consideramos realizar una PWA (Aplicación Web Progresiva, por sus siglas en inglés) con Vue, pero consideramos que no sería la mejor opción para el desarrollo de aplicaciones móviles. Si buscábamos hacer prototipos de aplicaciones, pensamos que lo mejor sería contar con una aplicación propia que nos permitiera alcanzar ese objetivo.

Finalmente llegamos a la conclusión que la mejor alternativa sería utilizar **React Native**, un framework de código abierto desarrollado por Facebook para aplicaciones móviles.

Dentro de las principales ventajas, React Native cuenta con muy buena documentación actualizada, mantenimiento constante, librerías para el control de gestos e imágenes con vastos años de experiencia y funcionamiento adecuado, dos puntos claves para el desarrollo de nuestra aplicación de digitalización de wireframes en papel a prototipos con navegación.

6.2 Tecnologías utilizadas propiamente para este desarrollo

En esta sección serán presentados los diferentes lenguajes, tecnologías y herramientas utilizadas para el desarrollo de nuestra aplicación de manipulación de wireframes y digitalización.

6.2.1 React

React[13] Es una biblioteca de JavaScript para construir interfaces de usuario. Esta biblioteca te permite crear interfaces de usuario interactivas de forma sencilla. Se

[13] https://en.reactjs.org/

encargará de actualizar y *renderizar* de manera eficiente los componentes correctos cuando los datos cambien. La lógica de los componentes está escrita en JavaScript[14].

React puede también *renderizar* desde el servidor usando Node, así como potencializar aplicaciones móviles usando React Native[15].

6.2.2 React Native

React Native combina las mejores partes del desarrollo nativo con React, la biblioteca de JavaScript para crear aplicaciones reales nativas para iOS y Android. Los componentes visuales creados corren directamente sobre las plataformas móviles nativas. Permite desarrollar aplicaciones cross-platform.

6.2.3 NodeJS

Es un entorno JavaScript del lado del servidor basado en eventos. Es una librería y entorno de ejecución de entrada/salida asíncrona, ejecutada sobre el intérprete de JavaScript. NodeJS[16] permite generar un sistema escalable y consistente que soporta conexiones intermitentes.

Propone un modelo con un único hilo de ejecución, utilizando E/S asíncronas que pueden ejecutarse en forma concurrente hasta cientos de miles sin condicionar el rendimiento general. Este diseño es eficaz en aplicaciones altamente concurrentes.

6.3 Arquitectura de la aplicación

El objetivo de esta sección es brindar información general acerca de cómo fue realizado el desarrollo de la aplicación MWP. Brindaremos detalles técnicos acerca de su construcción para facilitar la comprensión de aquellas personas que deseen continuar investigando sobre la digitalización de prototipos en papel.

6.3.1 Organización del código

Para poder comprender cómo es el funcionamiento técnico de la aplicación vamos a necesitar entender cómo se estructuran los diferentes módulos, como así

---

[14] https://www.javascript.com/
[15] https://reactnative.dev/
[16] https://nodejs.org/es/

también para qué sirven y qué información contiene cada uno de ellos. Existen muchas formas para organizar el código de una aplicación, pero en este caso hemos elegido agrupar los módulos de acuerdo a la funcionalidad que ofrecen.

Es por eso que decidimos organizar los módulos del proyecto en diferentes tipos y categorías. Por un lado, tenemos los módulos que son utilizados para representar diferentes componentes y elementos de la interfaz gráfica y por otro lado tenemos aquellos utilizados para gestionar la lógica general de la aplicación.

El módulo de la interfaz de usuario está conformado por diferentes pantallas y componentes personalizados complejos que creamos para el desarrollo de la aplicación. Entre los componentes personalizados se encuentran todos aquellos que sirven para representar cada uno de los widgets dentro de la previsualización o de las pruebas de usuario.

Por otra parte, dentro de los módulos de la lógica de la aplicación se encuentran 3 navigators, accesos directos, diferentes (galería, proyectos y previsualización) que nos ayudan a gestionar la navegación entre las diferentes pilas, donde para cada una de las opciones del menú tenemos un navegador diferente que nos facilita la gestión de navegación independiente en cada una de las pantallas.

A su vez, además de los *navigators* también contamos con módulos que nos ayudan a manejar la lógica de los datos de la aplicación. Estos datos, que muchas veces se comparten entre diferentes pantallas son gestionados utilizando el patrón de diseño Redux[17] el cual se encarga de administrar la información que se comparte entre las diferentes pantallas de la aplicación, y de esta forma permitir que exista la comunicación entre las distintas pantallas.

Por otra parte, también creamos diferentes clases de modelos que fueron de gran utilidad para poder darle un tipo a los distintos elementos que se crean tales como proyectos, wireframes, pruebas de usabilidad, la representación de los widgets, entre otros.

Por último, también se desarrollaron algunos servicios que nos permitieron generalizar ciertas partes de la lógica. Dichos servicios nos permiten encapsular algunas partes de la lógica de la aplicación en clases independientes que brindan determinadas utilidades al resto de los archivos mencionados previamente. Entre los servicios principales, podemos nombrar al servicio encargado de gestionar el guardado

[17] https://redux.js.org/

de las diferentes entidades de modelo o aquellos relacionados con la gestión de imágenes, proyectos, etc.

6.3.2 Flujo de información e interacción entre pantallas y componentes

Tal como mencionamos en la sección anterior, para desarrollar la aplicación hicimos uso de diferentes tipos de archivos tales como *navigators*, clases de modelo y archivos necesarios para implementar el patrón de diseño Redux, para así poder gestionar la lógica de datos de la aplicación.

La intención de esta sección es ampliar y brindar mayor nivel de detalle acerca de cómo es tratada la información y la comunicación entre las diferentes pantallas, tal como se detalla en la Figura 29.

Mientras los usuarios dan uso a la aplicación, existen muchas opciones donde es posible modificar dinámicamente la información. Cuando un usuario crea un nuevo proyecto o wireframe, carga los metadatos de un wireframe etiquetando sus diferentes componentes, desarrolla una nueva prueba de usabilidad, entre otras opciones, la información presentada en la aplicación cambia y es actualizada de manera dinámica.

Ante cada una de las acciones comentadas previamente, es posible que necesitemos actualizar la información en más de una pantalla. Ante cada uno de estos cambios, la información se guarda en el almacenamiento del dispositivo para tenerla disponible y permitir su uso entre las diferentes sesiones.

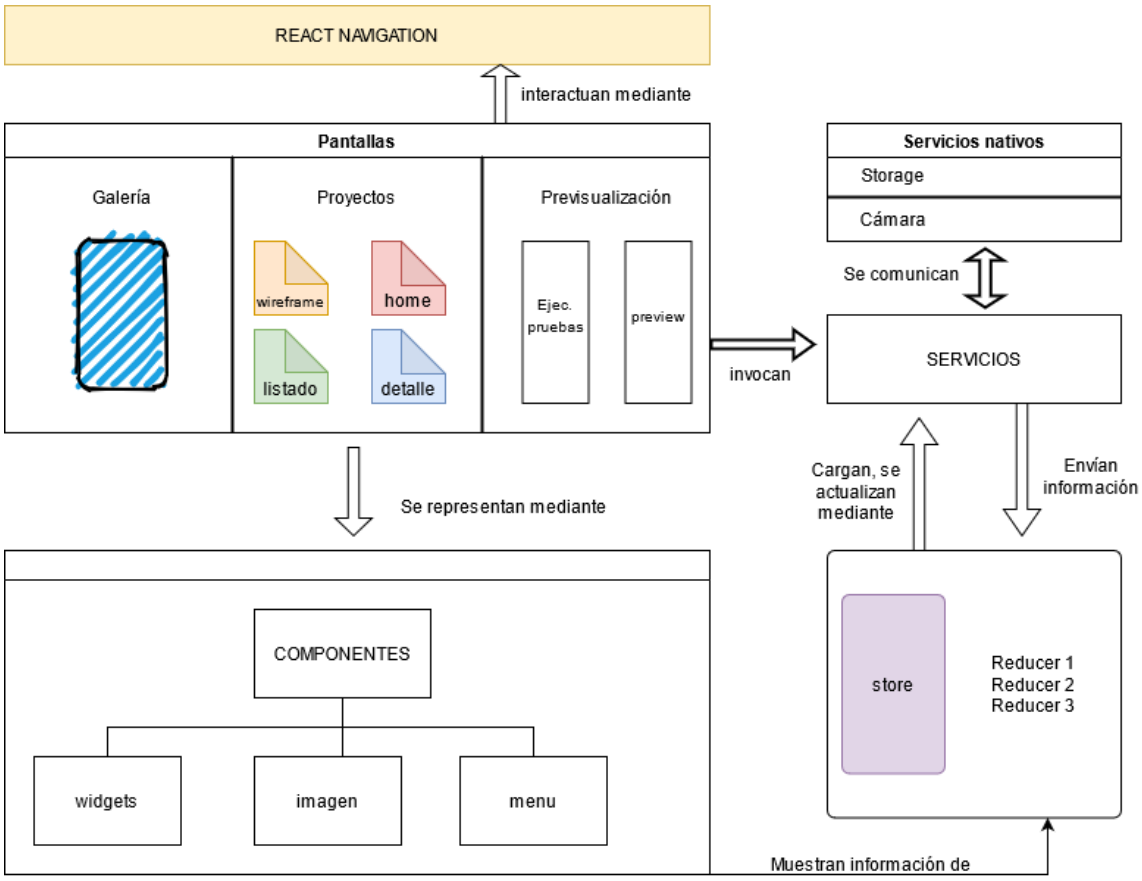

Figura 29 Diagrama sobre el flujo de información en la aplicación.

Sin embargo, frente a la necesidad de tener que compartir la información dinámicamente entre las diferentes pantallas se optó por utilizar Redux y de esta forma poder tener la información dinámica durante una misma sesión.

Para esto, se implementaron 5 reducers diferentes (funciones que nos permiten modificar el estado global de la aplicación), los cuales administran la información de los elementos de la galería de imágenes, los datos de todos los proyectos o de la previsualización y pruebas de usabilidad. Mediante diferentes acciones de Redux se mantiene la información actualizada dentro de la misma sesión pudiendo ser actualizada dinámicamente ante diferentes acciones del usuario.

Reducer	Usos
Gallery	La aplicación cuenta con una galería que nos permite guardar imágenes de aquellos componentes o interfaces visuales que puedan servir como motivación para el desarrollo de los proyectos. Este reducer gestiona la carga y representación de las imágenes de la galería.
Projects	Está relacionado con la gestión y administración de los diferentes proyectos dentro de la aplicación. Se encarga de la representación de los mismos en el listado de proyectos, guardado de proyectos nuevos, creación y actualización de los mismos. Gestiona la modificación de los atributos de cada uno de los proyectos tales como las secciones o wireframes que contiene cada uno.
Preview	Es el encargado de generar y controlar la previsualización de los diferentes wireframes ya sea durante la ejecución de las pruebas de usabilidad o bien en la pantalla de previsualización disponible para los desarrolladores y diseñadores. Genera el listado completo de componentes a presentar en la previsualización junto con sus respectivas propiedades.
Current Project	Para favorecer la rápida interacción de los usuarios con la aplicación se creó este reducer que nos permite ejecutar las diferentes acciones de los usuarios sobre un determinado proyecto de forma más rápida a la que tendríamos en comparación con el tiempo necesario que necesitamos para guardar la información de cada uno de los cambios generados en los proyectos. De esta forma, el usuario percibe una experiencia más fluida durante la utilización de la aplicación mientras que las acciones más costosas se ejecutan en segundo plano para evitar demoras en la respuesta de la app.
Current wireframe	Este último reducer cuenta con unas funciones muy

parecidas a las del reducer anterior, solamente que, en este caso, en lugar de administrar los cambios en un proyecto lo hace a nivel wireframe para mejorar la edición de los mismos de forma fluida.

Se crearon acciones tales como la creación de un nuevo proyecto, la actualización de un wireframe, los cambios en las diferentes pruebas de usabilidad o bien el estado de la previsualización o ejecución de las pruebas de usabilidad.

De esta manera, mediante la implementación de Redux se logró actualizar el contenido desde diferentes lugares de la aplicación manteniendo un estado consistente entre las diferentes pantallas, que representa en todo momento los últimos cambios realizados por el usuario.

Esto fue realizado de manera centralizada utilizando Redux, de una forma mucho más rápida que guardar los cambios en el almacenamiento del dispositivo para luego volver a cargar los mismos, demorando de esta forma la interacción con la aplicación debido a los altos tiempos de lectoescritura en el sistema de archivos de cada sistema operativo.

Por otro lado, además de administrar la información global de los diferentes proyectos, también mediante Redux, fue gestionada la información a mostrar en cada pantalla, en lugar de la utilización de parámetros de navegación convencionales. Donde en este último, realizar el flujo de información entre las pantallas podría resultar mucho más complejo.

6.3.3 Principales librerías utilizadas

Para poder realizar el desarrollo se han utilizado algunas librerías externas de terceros que nos permitieron avanzar sin necesidad de volver a implementar algunos de los comportamientos compartidos y existentes entre diferentes aplicaciones.

Como observamos en la Tabla 2, podemos diferenciar diferentes tipos de librerías. Por un lado, tenemos aquellas que se utilizaron como parte de la interfaz gráfica de la aplicación y por otro lado se agrupan las librerías que han sido utilizadas para otorgar cierta funcionalidad en la aplicación.

Como parte de las librerías utilizadas para el desarrollo de la interfaz gráfica podemos destacar la utilización de `@ui-kitten/components`[18], librería que nos proveyó muchos de los componentes visuales utilizados en la aplicación como así la librería `react-native-easy-grid`[19].

Esta última fue utilizada para la representación de los wireframes digitalizados. Mediante esta librería conseguimos representar los diferentes componentes marcados en cada uno de los wireframes de manera correcta representando la estructura y permitiendo que la grilla se pueda escalar correctamente en diferentes tamaños de pantalla.

Dentro de las librerías que otorgaron funcionalidad a la aplicación podemos diferenciar aquellas que gestionan el flujo de la información o la navegación entre pantallas. Entre estas se encuentran `@react-navigation`[20], librería que nos facilitó la navegación entre las diferentes pantallas de la aplicación.

Si bien la librería anterior nos permite enviar información entre las diferentes pantallas al momento de realizar navegación, muchas veces estos mecanismos resultan incompletos o muy complejos para utilizarlos. Para el caso de esta aplicación, si quisiéramos poder enviar y modificar información desde diferentes pantallas nos resultaría demasiado complejo.

Para poder mejorar el flujo de información entre los diferentes componentes y pantallas hemos optado por utilizar la librería `react-redux`[21] que nos permite la utilización del patrón de diseño Redux dentro de aplicaciones desarrolladas en React o React Native.

Una vez resuelta la navegación y el flujo de información entre las pantallas es necesario poder guardar esta información de manera persistente ya que react-redux solo mantiene la información dentro de la misma ejecución. Una vez que cerramos la aplicación esta información se pierde. Por este motivo hemos elegido utilizar la librería `@react-native-community/async-storage`[22] que nos permite guardar

---

[18] https://akveo.github.io/react-native-ui-kitten/docs/components/components-overview
[19] https://www.npmjs.com/package/react-native-easy-grid
[20] https://reactnavigation.org/
[21] https://react-redux.js.org/
[22] https://github.com/react-native-async-storage/async-storage

información de la aplicación de manera persistente dentro del almacenamiento del dispositivo.

Principales librerías utilizadas para el desarrollo de la aplicación MWP	
Librerías de uso general	• @react-navigation • @ui-kitten/components
Librerías para gestionar el flujo de información	• react-redux • @react-native-community/async-storage
Previsualización y pruebas de usabilidad	• react-native-easy-grid • @react-native-firebase/analytics
Captura, edición y markup de wireframes	• @react-native-community/image-editor • react-native-image-crop-picker • react-native-image-zoom-viewer

Tabla 2 Principales librerías utilizadas para el desarrollo de la aplicación MWP

Por otra parte, al ser una aplicación cuyo componente principal son las imágenes de los wireframes, también utilizamos librerías para trabajar con las capturas de los wireframes. En este grupo se encuentran `react-native-image-crop-picker`[23], librería que nos permite tomar imágenes desde la cámara del dispositivo o de la galería de imágenes guardadas en el teléfono.

Mediante esta librería no solo obtenemos las imágenes a usar, sino que también, podemos ofrecer una edición de las mismas para recortar solamente la parte que nos interesa, encuadrar y rotar las imágenes para que la parte seleccionada solamente comprenda a los wireframes y no otros elementos.

Continuando dentro de este mismo grupo también se encuentran `@react-native-community/image-editor`[24] que nos facilita el recorte de cada uno de los componentes marcados en los prototipos para mostrar en el formulario de edición de los componentes y `react-native-image-zoom-viewer`[25].

Con esta última librería desarrollamos la vista del wireframe para la edición de los componentes ya que nos permite visualizar una imagen y a su vez poder hacer zoom sobre ella para poder etiquetar aquellos componentes más pequeños. De esta forma,

[23] https://github.com/ivpusic/react-native-image-crop-picker
[24] https://www.npmjs.com/package/@react-native-community/image-editor
[25] https://www.npmjs.com/package/react-native-image-zoom-viewer

cuando es necesario etiquetar un componente como un ícono, es posible ampliar la imagen del wireframe para marcar todos los componentes que la conforman.

Por último, para poder implementar las diferentes métricas de usabilidad que se obtienen al realizar las pruebas de usabilidad con diferentes usuarios, fue necesario utilizar algún servicio especializado que nos permitiera capturar y procesar dicha información. Por este motivo, elegimos utilizar los servicios de Google Analytics[26]. Sumado a esto, también utilizamos la librería `@react-native-firebase/analytics`[27] la cual nos permitió conectar nuestra aplicación, con los servicios de Google Analytics para poder enviar los resultados de las diferentes métricas a evaluar.

Listado de las principales librerías utilizadas:

1. React-redux
2. @react-native-community/async-storage
3. @react-native-community/image-editor
4. @react-native-firebase/analytics
5. @react-navigation
6. @ui-kitten/components
7. react-native-easy-grid
8. react-native-image-crop-picker
9. react-native-image-zoom-viewer

6.4 Descripción técnica de las funcionalidades principales de la aplicación

Luego de brindar detalles generales sobre el lenguaje en el cual se desarrolló la aplicación y dar una noción sobre los conceptos relacionados con la arquitectura de la app, consideramos importante poder dar algunos ejemplos de aquellas funcionalidades más distintivas de nuestra aplicación. Por este motivo, en esta subsección vamos a explicar en mayor detalle cómo se realizaron las diferentes

[26] https://analytics.google.com
[27] https://www.npmjs.com/package/@react-native-firebase/analytics

funcionalidades necesarias para etiquetar y representar los prototipos en papel generados por los usuarios.

6.4.1 Etiquetado de cada widget en los wireframes

Uno de los principales desafíos al momento de realizar el desarrollo de la aplicación, fue encontrar la mejor forma de etiquetar los diferentes componentes de los wireframes.

Existen muchas formas diferentes para organizar la interfaz de usuario de una aplicación y estas formas de organizar la información han ido variando y mejorando a lo largo de los años. Pero el problema no fue solamente etiquetar los componentes, sino también encontrar la forma en la cual cada uno de esos componentes etiquetados pudiera ser representado de manera consistente con el prototipo de baja fidelidad (dibujado en papel). En este caso la problemática era doble: cómo etiquetar y cómo representar esos componentes.

Si el wireframe cuenta con una estructura y un determinado orden y posición de los componentes, es necesario poder respetar esa organización para que el prototipo digitalizado pueda ser fiel al dibujo realizado por el usuario. Por estos motivos necesitamos evaluar posibilidades que pudieran realizar las dos funciones, por un lado, capturar los widgets correctamente y que a la vez sean capaces de representar esos widgets de manera tal que pudieran ser renderizados en la previsualización.

Para conseguir esto hemos evaluado 3 diferentes posibles alternativas para realizar la selección de los componentes. Entre ellas se encontraron:

1. Etiquetar la imagen del wireframe mediante gestos de pantalla en los lugares donde estuvieran los widgets.
2. Realizar diferentes recortes en la imagen para obtener la posición de cada uno de los componentes.
3. Desarrollar un sistema de grilla que nos permita ir delimitando las diferentes partes del wireframe.

Cualquiera de las 3 alternativas podría haber funcionado correctamente para marcar y etiquetar los diferentes componentes, sin embargo, las dos primeras presentan una serie de desventajas que son necesarias tener en cuenta dado que

afectarían directamente la representación de la previsualización en las próximas etapas.

En una primera instancia habíamos optado por utilizar el primer enfoque y capturar los widgets mediante marcado con el *touch* en la pantalla sobre los diferentes widgets. Sin embargo, cuando quisimos avanzar en la representación nos encontramos con muchos problemas dada la información recolectada, ya que al seleccionar los componentes disponíamos únicamente de coordenadas.

Si bien la utilización de coordenadas es ampliamente utilizada, consideramos que resultaría difícil poder crear diseños responsivos que se adapten a las diferentes pantallas de los distintos dispositivos. Al momento de desarrollar esta tesina no existe ningún tipo de estándar sobre las características que deban tener las pantallas de los dispositivos móviles. Entre la amplia oferta de dispositivos podemos encontrar que muchos dispositivos tienen relaciones de aspecto *(Aspect ratio en inglés)* bastante diferentes. Es decir, la proporción entre el ancho y alto de la pantalla no es uniforme. A su vez la densidad de píxeles por pulgada puede ser muy diferente entre varios teléfonos.

Esta heterogeneidad en las características de las pantallas nos limita mucho la posibilidad de calcular la posición de los widgets en base a posiciones absolutas basadas en coordenadas en un dispositivo determinado. Si utilizamos un determinado dispositivo para etiquetar el widget y luego queremos visualizar el widget digitalizado en otro celular, esto puede resultar en representaciones muy diferentes.

No es lo mismo trabajar con un dispositivo con una alta o baja densidad de píxeles por pulgada, en donde las medidas de los componentes representados en pantalla pueden diferir mucho dependiendo de la densidad que cada dispositivo tenga. Sumado a esto, si consideramos que los dispositivos tienen diferentes relaciones de aspecto entre el alto y ancho, utilizar medidas absolutas puede traer muchos problemas, no permitiendo que los wireframes se representen siempre correctamente.

Con los primeros dos enfoques podemos obtener un conjunto de coordenadas **X** e **Y** para cada uno de los puntos que delimitan a los widgets. A su vez también podemos obtener las medidas de alto y ancho de cada componente. Sin embargo, como pudimos notar anteriormente, contar con medidas absolutas no nos permite poder desarrollar interfaces responsivas que se adapten correctamente a las diferentes resoluciones y relaciones de aspecto de las pantallas disponibles en el mercado.

Sumado a esto, conseguir desarrollar prototipos que se adapten a las pantallas partiendo de coordenadas resultaría muy complejo. La precisión con la que cada uno de los widgets marcados baja considerablemente. Las posiciones muchas veces pueden tener errores, lo cual puede traer muchos problemas. Si a la baja precisión de las coordenadas obtenidas sumamos también la complejidad de calcular la posición responsiva de cada elemento la representación podría diferir mucho de lo que el usuario intentó representar en su prototipo en papel.

Por los motivos detallados, decidimos optar por la última alternativa y desarrollar un sistema de grilla que nos permita calcular la posición de los widgets en pantalla. Si bien esta es la solución más compleja de implementar es la que nos resultaría más conveniente para poder etiquetar los componentes ya que no solo nos brinda información sobre la posición de los elementos, sino que también nos permite estructurar la representación de los mismos en filas y columnas.

6.4.2 Sistema de grilla para etiquetar los widgets

Tal como se mencionó en 6.4.1 , para poder etiquetar los diferentes widgets o componentes dentro de la aplicación se optó por desarrollar un sistema de guías verticales y horizontales que nos permitieran marcar los diferentes componentes dentro de la imagen. Por ejemplo, un wireframe puede estar conformado por un formulario, dos botones “Cancelar” y “Aceptar”, también podría contener una imagen o cualquier otro componente para la interfaz gráfica.

Para poder desarrollar la grilla para etiquetar los diferentes componentes, a lo largo del proceso de desarrollo necesitamos responder a algunos interrogantes que fueron surgiendo. En las siguientes subsecciones se explicarán detalladamente.

6.4.2.1 Representación de las líneas de la grilla y de los diferentes componentes

Para la representación de la grilla que marca la posición de cada uno de los componentes implementamos un mecanismo que nos permite etiquetar de manera sencilla cada uno de los componentes en la interfaz.

Teniendo en cuenta la necesidad de dar soporte a wireframes con elementos muy pequeños tales como íconos, fue necesario pensar una solución con la opción de

acercar y alejar las imágenes de los prototipos de forma tal que se puedan etiquetar elementos pequeños o bien una cantidad grande de widgets en el mismo wireframe.

Para poder resolver esta problemática se optó por utilizar la librería `react-native-image-zoom-view` que nos permitió ampliar y disminuir el tamaño de los wireframes. A su vez, también tiene la capacidad de representar capas superiores por encima de la imagen que nos dan la posibilidad de representar las líneas de la grilla y los widgets boxes (cajas de colores que representan a cada uno de los widgets etiquetados).

Esta librería también nos informa de los diferentes cambios tanto en el zoom como en la posición de la imagen. De esta forma, conociendo el zoom y desplazamiento de la imagen, podemos calcular las posiciones relativas de cada una de las líneas o interacción del usuario con sus prototipos en papel. Esto nos sirve para poder brindar mayor flexibilidad a la aplicación y para etiquetar de manera correcta cada uno de los widgets necesarios.

Para cada línea horizontal, vertical o widgetbox se calculan sus posiciones relativas dentro del prototipo en base al porcentaje de ubicación vertical u horizontal, así de esta forma podemos organizar en las próximas etapas la representación de los prototipos.

6.4.2.2 Como marcar componentes pequeños en los wireframes

Para poder etiquetar componentes pequeños decidimos utilizar la librería `react-native-image-zoom-view`. Al poder hacer zoom sobre las imágenes de los prototipos, se pueden etiquetar grandes cantidades de contenido en una misma imagen.

En esta nueva etapa, habiendo finalizado el *grid*, procedemos a seleccionar cada componente que deseemos representar. En función de la posición y zoom de la imagen tocaremos en diferentes partes de la misma. Conociendo la posición en la pantalla en la que el usuario hace el marcado, en conjunto con la posición de la imagen es posible determinar exactamente en qué punto de la imagen está presionando el usuario.

En función del punto (X, Y) de la imagen que el usuario toque, podremos determinar cuáles son las guías verticales y horizontales más cercanas. De esta forma, delimitaremos cada uno de los componentes pudiendo distinguir aquellas áreas donde debamos representar algún widget de las que no necesitemos representar ninguno.

6.4.3 Etiquetado de las diferentes propiedades de cada widget

Habiendo completado la selección de cada uno de los widgets a representar, es necesario poder establecer las propiedades que cada uno de los widgets necesita. Una vez finalizada la etapa del establecimiento de propiedades, tendremos acceso a la previsualización del prototipo o podremos realizar las pruebas de usabilidad disponibles.

Para conseguir eso se necesita que el usuario realice la personalización de cada widget de acuerdo a las necesidades del prototipo a representar. Cada uno de los widgets marcados tendrá diferentes propiedades que dependen del tipo seleccionado. No son las mismas propiedades las que se requieren para representar un *select* que una entrada de texto, un párrafo o una imagen.

De acuerdo al tipo de widget el usuario podrá personalizar ciertas opciones como podrían ser la selección de una imagen determinada o agregar opciones a un checkbox, entre otras disponibles.

6.4.4 Representación de los wireframes digitalizados

Una vez finalizado el etiquetado de los diferentes componentes que describimos en el punto anterior, podremos pasar a representar cada uno de nuestros prototipos en papel en prototipos manipulados digitalmente que nos permitan, ya sea, validar los requerimientos de una aplicación o realizar pruebas de usabilidad para poder comprobar la opinión de posibles usuarios finales basados en datos empíricos.

Con el etiquetado del paso anterior, obtenemos una serie de parámetros que nos servirán para la representación. En concreto, para poder realizar la previsualización contamos con algunos datos que se repiten para todos. Estos datos son: la posición en la interfaz, el tipo de widget y algunas propiedades personalizables que son diferentes dependiendo el tipo de widget seleccionado.

Como posición de cada elemento obtenemos 4 medidas, estas medidas representan en qué porcentaje de la imagen del prototipo se sitúa cada uno de los widgets. De esta forma, conociendo cada uno de los límites del widget (posición izquierda, derecha, arriba y abajo) podremos determinar en qué lugar de la interfaz debería ser representado.

Para esto se ordena cada uno de los widgets en el mismo sentido en el cual las personas de habla hispana leen un texto, es decir, de arriba hacia abajo y de izquierda a derecha, tomando como primer criterio de ordenación el orden vertical por sobre el horizontal.

Teniendo cada uno de los widgets ordenados de acuerdo a la posición que ocupa en la pantalla, procederemos a agruparlos en filas y de esta forma representaremos cada una de ellas con sus respectivas columnas. De esta manera conseguiremos representar cada widget en el orden correcto pudiendo ajustar su posición según las necesidades de cada uno de los dispositivos en los cuales se ejecute la aplicación.

6.4.5 Implementación de las métricas de usabilidad

En base a las métricas de usabilidad definidas en la sección 2.2, fue necesario implementar cambios en la aplicación para poder medir y hacer seguimiento del comportamiento de los usuarios durante la participación en las pruebas de usabilidad. Así, de esta forma, recolectamos información que nos ayude a realizar las mediciones y obtener resultados empíricos luego de la ejecución de las pruebas con usuarios finales.

Para ello, tal como vimos en 5.4.1 fue necesario definir una serie de eventos que nos permitieran entender cómo los usuarios interactúan con los prototipos y obtener determinados parámetros e información a medida que los usuarios participan de las pruebas de usabilidad con los prototipos digitalizados.

Cada tipo de evento tiene tanto un conjunto de datos generales a recolectar en cada uno de ellos como así también algunos parámetros adicionales específicos de cada tipo de evento. Entre los *parámetros generales* se encuentran:

- **project_id**: Número único que identifica de manera unívoca cada proyecto.
- **testing_id**: Número único que identifica de manera unívoca cada una de las pruebas de usabilidad.
- **project_name**: Nombre del proyecto para el cual se está realizando la prueba
- **testing_name**: Nombre de la prueba de usabilidad que se está analizando.
- **first_time**: Indica si el usuario ya completó o no la prueba de usabilidad. Nos sirve para poder comparar los tiempos de ejecución de las pruebas en base a la experiencia de los usuarios habiendo realizado la prueba anteriormente.

- **current_screen_id**: Número único que identifica a cada pantalla de cada prototipo. Nos sirve para determinar en qué pantalla se encuentra el usuario en cada evento.
- **current_screen_name**: Nombre del wireframe que el usuario se encuentra probando.
- **current_timestamp**: Marca temporal que nos permite hacer un seguimiento de la cronología de los eventos registrados por los usuarios y de esta forma conocer cuáles fueron los pasos que cada uno de ellos realizó.
- **min_screen_count**: Cantidad mínima de pantallas que el usuario debe visitar para poder finalizar la prueba. Este número se define al momento de crear la prueba de forma manual.

Parámetros específicos para el evento de navegación:

- **destination_route_id**: Identificador numérico único de cada pantalla que indica el número de la próxima pantalla a visitar.
- **destination_route_name**: Nombre de la pantalla a la cual el usuario intenta navegar.
- **total_screen_time**: Tiempo total en la última pantalla antes de la navegación a la siguiente.

Parámetros específicos del evento de finalización de la prueba:

- **total_testing_time**: Valor que nos indica la cantidad total de tiempo que el usuario estuvo participando de la prueba. Está indicado en milisegundos.
- **total_different_screens_count**: Cantidad de pantallas diferentes por las cuales el usuario pasó antes de finalizar la prueba. Solamente se cuentan las pantallas diferentes, es decir, si el usuario visitó 2 o más veces la misma pantalla esta deja de contabilizarse.
- **total_screen_count**: Cantidad total de pantallas que el usuario visitó sin importar si fueron o no visitadas previamente. Si el usuario visita una pantalla 2 veces se contará en este valor cada uno de las veces que haya pasado por la pantalla.
- **testing_status**: Valor que representa el resultado de la prueba realizada. Nos indica la causa por la cual fue finalizada la misma, pudiendo considerarse como motivo de finalización que el usuario haya completado correctamente la misma, haya decidido abortar la prueba o finaliza por pasar el tiempo máximo de la prueba.

Parámetros específicos del evento de calificación de la prueba:

- **facilidad_uso**: Número del 1 al 5 que representa cuán sencillo de usar resultó el prototipo. Mientras mayor sea el valor obtenido, más fácil considerarán los usuarios que fue completar la prueba.
- **nivel_satisfacción**: Número del 1 al 5 que representa cuán satisfechos quedaron los usuarios en la prueba realizada. Mientras mayor sea el valor obtenido, más satisfechos quedaron los usuarios que hayan participado de la prueba.
- **Comentario**: Texto libre donde el usuario puede incluir sus opiniones sobre la prueba realizada.

Capítulo 7: Resultados obtenidos, conclusiones y trabajo a futuro

En este capítulo, buscaremos poner al lector en tema sobre los avances conseguidos luego del desarrollo de la tesina. En un principio, al momento de comenzar con la misma se plantearon muchos objetivos e ideales que quisimos resolver o sobre los que queríamos contribuir. Sin embargo, en la práctica, luego de adentrarnos mejor en el tema e investigar más en profundidad es posible ir mejorando dichos objetivos de acuerdo a los resultados que conseguimos durante el proceso.

El desarrollo de la tesina involucra muchas actividades diferentes. A lo largo de las mismas uno va aprendiendo sobre el tema que planteó en un principio y con este aprendizaje se pueden refinar las necesidades.

También mencionaremos las conclusiones definitivas que logramos obtener de todo el desarrollo de la tesina de Manipulación de Prototipos en Papel. Además, realizamos un listado de las posibles funcionalidades y características que agregarían valor a la aplicación y que se desprenden de la presente tesina como trabajo a futuro.

7.1 Resultados obtenidos

Una de las principales actividades para el desarrollo de la tesina fue realizar una investigación, en gran profundidad, sobre el tema a desarrollar ya que es de vital importancia conocer los trabajos relacionados para no repetir los mismos. Esto también nos ayudó a conocer mejor qué otros avances existían en el tema a trabajar.

Por otra parte, al no encontrar ninguna aplicación que cumpliera nuestras necesidades fue necesario el desarrollo de una aplicación propia que ayude a mejorar las problemáticas planteadas en la motivación. Esto fue un gran desafío para el cual necesitamos hacer distintos tipos de pruebas de conceptos para encontrar la mejor forma de implementar la aplicación como se explicó en el capítulo 6.

Por último, también hemos necesitado hacer diferentes pruebas que nos permitieron evaluar si la aplicación desarrollada podía ser aplicada correctamente para resolver las problemáticas planteadas de usabilidad y elicitación de requerimientos preestablecidas en el comienzo.

Luego de este todo este trabajo podemos afirmar que hemos logrado varios avances en materia de pruebas de usabilidad y elicitación de requerimientos. Entre los resultados principales se encuentran:

1. Se logró desarrollar una aplicación que permite digitalizar prototipos en papel. A través del procesamiento de las imágenes, selección de los widgets, posterior análisis y asignación de las propiedades podemos obtener una previsualización de los wireframes digitalizados con sus correspondientes widgets.

2. Se consiguió crear una herramienta para evaluar usabilidad en etapas tempranas del desarrollo. Por otro lado, al hacer uso de la herramienta, podremos obtener resultados empíricos de la experiencia de usuario en las primeras etapas de diseño de productos de software, previniendo posibles problemas de usabilidad. A su vez, contribuimos a disminuir los errores en el desarrollo de productos de software al contar con prototipos digitalizados en la aplicación.

3. Mediante la utilización de la aplicación creada para la tesina, se consiguió una forma de poder centralizar las opiniones y sugerencias de posibles usuarios finales que participen de las pruebas de usabilidad en las etapas tempranas de desarrollo, incluso antes de realizar un desarrollo completo para poder obtener información de los usuarios. Esto resulta de mucha importancia ya que dependiendo de los resultados que arroje la evaluación de la usabilidad y calidad de software podremos obtener productos con mejor calidad.

   La opinión de los usuarios finales es importante para cualquier software ya que es lo que determinará el éxito de la aplicación. Durante el desarrollo de software es indispensable mantener e incluso, mejorar la satisfacción de los usuarios finales.

   La importancia del uso y análisis de las métricas reside en que nos aportan toda la información que necesitamos para llevar a cabo las medidas más adecuadas al proyecto.

4. A su vez, logramos crear una aplicación que ayude a facilitar los procesos de elicitación de requerimientos. Comprobamos que contar con una aplicación para digitalizar prototipos es útil durante el proceso de elicitación de requisitos y facilita la misma. Al poder digitalizar un wireframe en papel con simples pasos los clientes podrán ver si el wireframe digitalizado cuenta con todos los requerimientos esperados. Este proceso se simplifica ya que las modificaciones podrán ser debatidas y realizadas tan pronto como se cuente con el nuevo wireframe digitalizado.

7.2 Conclusiones

A lo largo de la tesina se realizó un análisis de las ventajas y desventajas de la utilización del paradigma *Mobile First* para aplicaciones móviles y web, rectificando que el paradigma *Mobile First* favorecerá en gran medida a estos sistemas, subsanando las problemáticas de diseño e implementación de las aplicaciones.

Se diseñó, desarrolló e implementó un sistema que permite la digitalización de wireframes en papel desde la captura de una imagen hasta, luego de un marcado de los widgets, la posibilidad de realizar la previsualización con navegación completa del mismo wireframe inicial que fue capturado y subido a la aplicación.

También podemos concluir que nuestra aplicación contribuye a la obtención de métricas de usabilidad de manera temprana. Así, se vuelve posible realizar ajustes en la aplicación que se desee construir durante el proceso de desarrollo. Por lo tanto, se podrá conseguir un mayor grado de éxito, focalizando en brindar una buena experiencia a los usuarios finales. Algunas de las métricas que son posibles de obtener son el nivel de dificultad y satisfacción de los usuarios, porcentaje de tareas completadas con éxito en el primer intento, tiempo empleado en completar una tarea, tiempo transcurrido en cada pantalla, entre otras.

Sin embargo, las muestras tomadas durante el desarrollo de la aplicación han sido realizadas en un grupo reducido de personas. Motivo por el cual, hay que tener en consideración que para que los resultados de la muestra sean realmente representativos, deberán realizarse con un número mayor de usuarios que pertenezcan a diferentes grupos sociales. De esta manera obtendremos una retroalimentación de muchas personas con gran agilidad.

7.3 Trabajos futuros

A continuación, se describen algunos de los posibles trabajos futuros que se desprenden de la presente tesina:

- Permitir la edición colaborativa de los wireframes en grupos de diferentes usuarios desde distintos dispositivos.
- Utilización de técnicas de detección automática de widgets y elementos en los mockups utilizando inteligencia artificial y deep learning.
- Ampliar las métricas que se pueden obtener con la aplicación.
- Continuar el desarrollo para migrar la aplicación móvil a una aplicación web o desktop.
- Ampliar la aplicación para digitalizar prototipos en papel para web.
- Añadir uno o más widgets al sistema adaptando la solución con el fin de poder utilizar los nuevos widgets en previsualizaciones futuras.

- Obtener los datos para mostrar en la previsualización desde fuentes externas tales como archivos o a través de la comunicación con una API que nos permita visualizar contenido dinámico dentro de la aplicación. De esta forma podríamos ampliar las pruebas para utilizar datos reales del dominio de cada aplicación.

- Permitir agregar comentarios sobre la previsualización de los wireframes digitalizados.

- Implementar un mecanismo que nos permita ejecutar de manera independiente las diferentes pruebas de usabilidad creadas en la aplicación. Así de esta forma podríamos permitir a un mayor número de usuarios participar de las mismas y como resultado obtendremos información más completa y fiable mediante la opinión de un número relevante de personas.

- Añadir soporte para más gestos haciendo uso de **RNGestureHandler**. *React Native Gesture Handler* proporciona una API de gestión de gestos nativa para crear las mejores experiencias táctiles posibles.
  Con esta biblioteca, los gestos no son controlados por el sistema de respuesta JS, sino que se reconocen y se rastrean en la UI (interfaz de usuario) haciendo que las interacciones táctiles y el seguimiento de gestos no solo sean fluidos, sino también confiables y deterministas.

Referencias

Antti Oulasvirta, Tye Rattenbury, Lingyi Ma, and Eeva Raita. Habits make smartphone use more pervasive. *Personal and Ubiquitous Computing*, 16(1):105–114, 2012.

Cooper, A. (2004). *The inmates are running the asylum:[Why high-tech products drive us crazy and how to restore the sanity]* (Vol. 2). Indianapolis: Sams.

Holzmann, C., & Vogler, M. (2012, August). Building interactive prototypes of mobile user interfaces with a digital pen. In *Proceedings of the 10th asia pacific conference on Computer human interaction* (pp. 159-168).

Huang, R., Long, Y., & Chen, X. (2016). Automatically Generating Web Page From A Mockup. In *SEKE* (pp. 589-594).

Kolko, J. (2010). *Thoughts on interaction design*. Morgan Kaufmann.

Landay, J. A. (1996). *Interactive Sketching for the Early Stages of User Interface Design* (No. CMU-CS-96-201). CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

Li, Y., Cao, X., Everitt, K., Dixon, M., & Landay, J. A. (2010, April). FrameWire: a tool for automatically extracting interaction logic from paper prototyping tests. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 503-512).

Lin, J., Newman, M. W., Hong, J. I., & Landay, J. A. (2000, April). DENIM: finding a tighter fit between tools and practice for Website design. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 510-517).

McCurdy, M., Connors, C., Pyrzak, G., Kanefsky, B., & Vera, A. (2006, April). Breaking the fidelity barrier: an examination of our current characterization of prototypes and an example of a mixed-fidelity success. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 1233-1242).

Murillo, D., & Herrera, J. (2015, July). Use of wireframes and mockup on the redesign of a university website using the methodology User Centered Design. Memorias del VII Congreso Iberoamericano de Telemática CITA2015.

Nielsen, J. (1994, April). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 152-158).

Snyder, C. (2003). *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann.

Davis, R. C., Saponas, T. S., Shilman, M., & Landay, J. A. (2007, October). SketchWizard: Wizard of Oz prototyping of pen-based user interfaces. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (pp. 119-128).

Seifert, J., Pfleging, B., del Carmen Valderrama Bahamóndez, E., Hermes, M., Rukzio, E., & Schmidt, A. (2011, August). Mobidev: a tool for creating apps on mobile phones. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (pp. 109-112).

Landay, J. A., & Myers, B. A. (2001). Sketching interfaces: Toward more human interface design. *Computer*, *34*(3), 56-64.

Nguyen, T. A., & Csallner, C. (2015, November). Reverse engineering mobile application user interfaces with remaui (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 248-259). IEEE.

Beltramelli, T. (2018, June). pix2code: Generating code from a graphical user interface screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (pp. 1-6).

Nielsen, J. (2000). Designing web usability.

Vijayan, J., & Raju, G. (2011). A new approach to requirements elicitation using paper prototype. *International Journal of Advanced Science and Technology*, *28*, 9-16.

Petrie, H., & Bevan, N. (2009). The Evaluation of Accessibility, Usability, and User Experience. *The universal access handbook*, *1*, 1-16.

DeMarco, T. (1986). *Controlling software projects: Management, measurement, and estimates*. Prentice Hall PTR.

Chang, E., & Dillon, T. S. (2006). A usability-evaluation metric based on a soft-computing approach. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *36*(2), 356-372.

Nielsen, J., "Why You Only Need to Test With 5 Users", Alertbox March 19, 2000, at http://www.useit.com/alertbox/20000319.html